EBOOK HYPERKB OS

AGI-Oriented HyperKnowledge Base for creating serious, simple, research-driven, applicable, pedagogical, visually coherent ebooks.

With Rewriting Engine, RAG/GraphRAG Layer, HyperSkills, HyperLoops, Destruction Courts, Marketplace Deployment, and a final reusable AGI OS kernel.

CONCEPT ART

Neo-abstract tech aesthetic, flat design, thumbnail-first title scale.

101 CLARITY

Simple expression without killing depth, mechanism, or evidence rank.

MOUSSAIF OS

MCD / QMCD / MCAI / HAKIM-MCD as generative lenses, not factual proof.

ZOUHAIR MOUSSAIF / MOUSSAIF KNOWLEDGE ARCHITECTURE - JULY 2026 - VERSION 2.0

Epistemic Safety Declaration

Operational frame

Layer	Required output
Available material	User prompt, prior HyperKB memory,
Directly sourced	Platform rules, accessibility standards,
Inferred	Operational architecture connecting
Theoretical synthesis	MCD, QMCD, MCAI, HAKIM-MCD,
Cannot claim	No guarantee of AGI, sales, compliance,

Use in practice

Available material: user prompt, prior Moussaif HyperKB memory, uploaded-file snippets, and current public sources on EPUB, accessibility, KDP, SEO, RAG, GraphRAG, UX, and LLM security. Directly sourced material is separated from inferred architecture. MCD, QMCD, MCAI, HAKIM-MCD, HyperSkills, HyperLoops, and Destruction Courts are author-level operating lenses, not established scientific facts. This PDF is a serious knowledge-system artifact, not legal, medical, publishing, or financial certification. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Executive OS Abstract

Operational frame

Layer	Required output
Core	The ebook is treated as four things at
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

The ebook is treated as four things at once: a learning route, a research artifact, a product, and a machine-readable knowledge system. A serious ebook does not merely inform; it changes the reader's next action, supplies retrieval anchors, teaches via examples and exercises, and repairs itself when evidence, platform rules, or reader outcomes contradict the first draft. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Domain Constitution

Operational frame

Layer	Required output
Core	Purpose: create serious, simple,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Purpose: create serious, simple, applicable, pedagogical ebooks that preserve depth while reducing friction. Scope: research, structuring, writing, rewriting, visual design, page filling, publishing, SEO, RAG/GraphRAG, agentic workflows, and evaluation. Exclusions: plagiarism, blind paraphrase, fake expertise, medical certainty without clinical sources, platform-rule guessing, and decorative theory that does not become a tool. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Ontological Status of Ebook Creation

Operational frame

Layer	Required output
Core	The domain is not a single object. It is a
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

The domain is not a single object. It is a process, system, network, language, business product, pedagogical mechanism, technical architecture, and AGI-oriented knowledge interface. Treating it only as writing kills deployment; treating it only as design kills evidence; treating it only as SEO kills trust; treating it only as AI generation kills authorship. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Central Research Question

Operational frame

Layer	Required output
Core	How can an ebook be engineered so that
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

How can an ebook be engineered so that each page teaches, proves, applies, retrieves, sells ethically, updates, and survives critique without becoming either shallow content marketing or unreadable academic density? Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Generative Kernel

Operational frame

Layer	Required output
Core	Reader transformation + source ledger +
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Reader transformation + source ledger + mechanism graph + 101 compression + visual hierarchy + exercises + publishing metadata + RAG chunks + destruction courts + version memory. This kernel can regenerate the whole system because every later section is one expansion of these ten operations. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Evidence and Confidence Layers

Operational frame

Layer	Required output
Core	Every claim receives a class: fact,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Every claim receives a class: fact, interpretation, model, tool, metaphor, hypothesis, proposal. Every class receives an epistemic rank: established, supported, plausible, contested, theoretical synthesis, hypothesis, speculative, unknown, or needs verification. The OS must not let a beautiful page disguise an unsupported claim. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Operational Glossary Protocol

Operational frame

Layer	Required output
Core	A concept enters the KB only if it has a
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

A concept enters the KB only if it has a name, definition, function, distinction, OS role, operational use, misuse risk, epistemic rank, related concepts, example, and test question. A weak glossary explains vocabulary; a strong glossary controls future action and retrieval. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Concept Graph

Page 10 / 700 - Operating contract - AGI-oriented knowledge base

Operational frame

Layer	Required output
Core	Core nodes: reader, promise, claim,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Core nodes: reader, promise, claim, source, mechanism, example, exercise, visual block, page architecture, rewrite operation, product metadata, retrieval chunk, graph relation, court, metric, update event. Core edges: defines, supports, contradicts, operationalizes, trains, retrieves, falsifies, updates. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Problem Graph

Operational frame

Layer	Required output
Core	Major problems: shallow AI content,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Major problems: shallow AI content, source opacity, unreadable density, white-space waste, visual incoherence, beginner-expert gap, unsafe mental-health advice, weak tech explanations, SEO-first thin content, platform rejection, hallucinated tool capability, and unmeasured learning outcome. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Mechanism Graph

Operational frame

Layer	Required output
Core	Each mechanism has input,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Each mechanism has input, transformation, mediator, moderator, threshold, feedback, output, failure condition, example, and test protocol. For example, 'simple expression' is not lower intelligence; it is a transformation that reduces extraneous load while preserving mechanism, distinction, and evidence rank. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

HyperSkills Overview

Operational frame

Layer	Required output
Core	A HyperSkill is not a small trick. It is a
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

A HyperSkill is not a small trick. It is a reusable capability that transfers across domains because it preserves the governing distinction. Here the governing distinction is: the page is accepted only when it produces a reader action, a source trace, a test, a tool, a diagram, a prompt, or a decision. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

HyperLoops Overview

Operational frame

Layer	Required output
Core	The loops are Kernel Extraction, Evidence
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

The loops are Kernel Extraction, Evidence Ranking, Concept Disambiguation, Problem Decomposition, Hypothesis Generation, Destruction and Repair, Tool Construction, Skill Training, Retrieval Optimization, Agentic Execution, Update and Versioning, and Research Program Generation. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Agentic Interfaces

Page 15 / 700 - Operating contract - AGI-oriented knowledge base

Operational frame

Layer	Required output
Core	Agents: Research Agent, Concept Agent,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Agents: Research Agent, Concept Agent, Evidence Agent, Critique Agent, Tool Agent, Skill Agent, Retrieval Agent, Graph Agent, Pedagogy Agent, Deployment Agent, Destruction Court Agent, and Versioning Agent. Each agent has allowed operations, forbidden operations, and evaluation criteria. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

RAG and GraphRAG Readiness

Operational frame

Layer	Required output
Core	Chunk types: concept, problem,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Chunk types: concept, problem, mechanism, tool, skill, loop, evidence, critique, example, hypothesis, update. Metadata: domain, section, concept, confidence, centrality, relation type, skill relevance, tool relevance, risk, version, and update status. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Tool Layer

Operational frame

Layer	Required output
Core	Core tools: Concept Kernelizer, Evidence
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Core tools: Concept Kernelizer, Evidence Ranker, Difference Extractor, MCD Evaluator, QMCD Model Superposition Map, Hypothesis Generator, Destruction Court Matrix, HyperSkill Distiller, HyperLoop Orchestrator, RAG Chunking Compiler, GraphRAG Relation Compiler, AGI-Readiness Index. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Destruction Courts

Operational frame

Layer	Required output
Core	Every output passes Hallucination Court,
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Every output passes Hallucination Court, Reductionism Court, False Novelty Court, Weak Operationalization Court, Agentic Failure Court, Publishing Compliance Court, and Reader Outcome Court. The court must produce a repair, not just criticism. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

AGI-Readiness Metrics

Operational frame

Layer	Required output
Core	Metrics are qualitative unless real data
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

Metrics are qualitative unless real data exists: Kernel Regeneration, Concept Operationality, Evidence Governance, Retrieval Precision, Graph Connectivity, HyperSkill Density, HyperLoop Maturity, Destruction Resistance, Agentic Deployment, Research Opening, Update Capacity, and MCD Score. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Table of Contents / Retrieval Map

Operational frame

Layer	Required output
Core	P1 p.21-65 Ebook Product Constitution. P2
Function	Turn the section into an operational route:
Failure mode	Long essay, glossary, or fake-deep
Update	Recheck sources, reader outcomes,

Use in practice

P1 p.21-65 Ebook Product Constitution. P2 p.66-110 Research Engine. P3 p.111-155 Pedagogical Simplicity. P4 p.156-200 Writing Engine. P5 p.201-245 Visual Coherence. P6 p.246-290 Rewriting Engine. P7 p.291-335 Technology Adapter. P8 p.336-380 Mental Health Adapter. P9 p.381-425 Marketing Adapter. P10 p.426-470 Publishing and SEO. P11 p.471-515 Agentic Pipeline. P12 p.516-560 RAG/GraphRAG. P13 p.561-605 Courts. P14 p.606-650 Scaling. Appendix p.651-700. Convert every idea into manuscript section, worksheet, prompt, checklist, diagram, RAG chunk, GraphRAG relation, product asset, court verdict, or memory update.

Mini-test

What is established? What is synthesis? What is the tool? What is the failure mode? What memory update remains after use? What would make the page safer, clearer, more operational, and easier to retrieve?

Operating law: knowledge -> operation -> skill -> loop.

OS actions

1. Declare source status and confidence rank.
2. Extract kernel before chapter expansion.
3. Use 101 clarity without deleting mechanisms.
4. Attach worksheet, prompt, table, or mini-test.
5. Run courts and update memory.

Page-fill memory

Dense pages need grammar: headline, mechanism, example, checklist, court, metadata, and update logic. No empty decorative waste; no text wall without hierarchy. The grid is not decoration: it is a cognitive scaffold that keeps the page readable while using space intensely. A page passes when it changes a decision, improves a manuscript, clarifies evidence, trains a skill, or makes the next retrieval more precise.

Source anchors for this OS: S1 W3C EPUB 3.3 / EPUB Accessibility | S2 W3C WCAG 2.2 | S3 Amazon KDP Help / Content Guidelines | S4 Google Search Central | S5 Lewis et al. RAG 2020 | S6 Microsoft GraphRAG

Reader Promise - Operating Card 01

Core proposition

Reader promise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader promise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader promise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader promise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader promise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader promise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Spine - Operating Card 01

Core proposition

Offer spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Use-Case Ladder - Operating Card 01

Core proposition

Use-case ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by use-case ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, use-case ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does use-case ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=use-case ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain use-case ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Audience Contract - Operating Card 01

Core proposition

Audience contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by audience contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, audience contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does audience contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=audience contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain audience contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Outcome Map - Operating Card 01

Page 25 / 700 - P1 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Outcome map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by outcome map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, outcome map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does outcome map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=outcome map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain outcome map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Chapter Economy - Operating Card 01

Core proposition

Chapter economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Stack - Operating Card 01

Core proposition

Asset stack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset stack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset stack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset stack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset stack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset stack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Trust Architecture - Operating Card 01

Core proposition

Trust architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Acceptance Rule - Operating Card 01

Core proposition

Page acceptance rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page acceptance rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page acceptance rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page acceptance rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page acceptance rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page acceptance rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Product Positioning - Operating Card 01

Core proposition

Product positioning is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product positioning.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product positioning becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product positioning hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product positioning; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product positioning? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Pricing Logic - Operating Card 01

Page 31 / 700 - P1 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Pricing logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pricing logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pricing logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pricing logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pricing logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pricing logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Delivery Bundle - Operating Card 01

Core proposition

Delivery bundle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by delivery bundle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, delivery bundle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does delivery bundle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=delivery bundle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain delivery bundle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Reader Onboarding - Operating Card 01

Core proposition

Reader onboarding is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader onboarding.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader onboarding becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader onboarding hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader onboarding; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader onboarding? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Field Boundary - Operating Card 01

Core proposition

Field boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by field boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, field boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does field boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=field boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain field boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Success Metric - Operating Card 01

Core proposition

Success metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by success metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, success metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does success metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=success metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain success metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Reader Promise - Operating Card 02

Core proposition

Reader promise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader promise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader promise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader promise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader promise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader promise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Spine - Operating Card 02

Core proposition

Offer spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Use-Case Ladder - Operating Card 02

Core proposition

Use-case ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by use-case ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, use-case ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does use-case ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=use-case ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain use-case ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Audience Contract - Operating Card 02

Core proposition

Audience contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by audience contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, audience contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does audience contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=audience contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain audience contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Outcome Map - Operating Card 02

Page 40 / 700 - P1 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Outcome map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by outcome map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, outcome map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does outcome map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=outcome map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain outcome map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Chapter Economy - Operating Card 02

Core proposition

Chapter economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Stack - Operating Card 02

Core proposition

Asset stack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset stack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset stack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset stack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset stack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset stack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Trust Architecture - Operating Card 02

Core proposition

Trust architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Acceptance Rule - Operating Card 02

Core proposition

Page acceptance rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page acceptance rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page acceptance rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page acceptance rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page acceptance rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page acceptance rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Product Positioning - Operating Card 02

Core proposition

Product positioning is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product positioning.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product positioning becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product positioning hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product positioning; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product positioning? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Pricing Logic - Operating Card 02

Core proposition

Pricing logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pricing logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pricing logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pricing logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pricing logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pricing logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Delivery Bundle - Operating Card 02

Core proposition

Delivery bundle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by delivery bundle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, delivery bundle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does delivery bundle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=delivery bundle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain delivery bundle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Reader Onboarding - Operating Card 02

Core proposition

Reader onboarding is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader onboarding.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader onboarding becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader onboarding hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader onboarding; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader onboarding? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Field Boundary - Operating Card 02

Core proposition

Field boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by field boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, field boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does field boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=field boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain field boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Success Metric - Operating Card 02

Core proposition

Success metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by success metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, success metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does success metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=success metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain success metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Reader Promise - Operating Card 03

Core proposition

Reader promise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader promise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader promise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader promise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader promise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader promise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Spine - Operating Card 03

Core proposition

Offer spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Use-Case Ladder - Operating Card 03

Core proposition

Use-case ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by use-case ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, use-case ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does use-case ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=use-case ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain use-case ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Audience Contract - Operating Card 03

Core proposition

Audience contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by audience contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, audience contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does audience contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=audience contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain audience contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Outcome Map - Operating Card 03

Core proposition

Outcome map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by outcome map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, outcome map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does outcome map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=outcome map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain outcome map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Chapter Economy - Operating Card 03

Core proposition

Chapter economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Stack - Operating Card 03

Core proposition

Asset stack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset stack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset stack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset stack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset stack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset stack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Trust Architecture - Operating Card 03

Core proposition

Trust architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Acceptance Rule - Operating Card 03

Core proposition

Page acceptance rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page acceptance rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page acceptance rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page acceptance rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page acceptance rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page acceptance rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Product Positioning - Operating Card 03

Core proposition

Product positioning is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product positioning.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product positioning becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product positioning hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product positioning; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product positioning? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Pricing Logic - Operating Card 03

Core proposition

Pricing logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pricing logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pricing logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pricing logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pricing logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pricing logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Delivery Bundle - Operating Card 03

Core proposition

Delivery bundle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by delivery bundle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, delivery bundle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does delivery bundle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=delivery bundle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain delivery bundle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Reader Onboarding - Operating Card 03

Core proposition

Reader onboarding is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader onboarding.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader onboarding becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader onboarding hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader onboarding; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader onboarding? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Field Boundary - Operating Card 03

Core proposition

Field boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by field boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, field boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does field boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=field boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain field boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Success Metric - Operating Card 03

Core proposition

Success metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by success metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, success metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does success metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=success metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain success metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Source Ecology - Operating Card 01

Core proposition

Source ecology is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source ecology.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source ecology becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source ecology hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source ecology; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source ecology? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Primary Source - Operating Card 01

Core proposition

Primary source is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by primary source.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, primary source becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does primary source hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=primary source; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain primary source? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Secondary Synthesis - Operating Card 01

Core proposition

Secondary synthesis is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by secondary synthesis.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, secondary synthesis becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does secondary synthesis hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=secondary synthesis; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain secondary synthesis? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Ledger - Operating Card 01

Core proposition

Claim ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Evidence Rank - Operating Card 01

Core proposition

Evidence rank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence rank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence rank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence rank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence rank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence rank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Citation Discipline - Operating Card 01

Core proposition

Citation discipline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by citation discipline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, citation discipline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does citation discipline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=citation discipline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain citation discipline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Uncertainty Map - Operating Card 01

Core proposition

Uncertainty map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by uncertainty map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, uncertainty map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does uncertainty map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=uncertainty map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain uncertainty map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Contradiction Table - Operating Card 01

Core proposition

Contradiction table is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contradiction table.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contradiction table becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contradiction table hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contradiction table; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contradiction table? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Research Question - Operating Card 01

Core proposition

Research question is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research question.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research question becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research question hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research question; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research question? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Mechanism Extraction - Operating Card 01

Core proposition

Mechanism extraction is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism extraction.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism extraction becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism extraction hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism extraction; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism extraction? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Source Triangulation - Operating Card 01

Core proposition

Source triangulation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source triangulation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source triangulation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source triangulation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source triangulation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source triangulation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Authority Vs Function - Operating Card 01

Core proposition

Authority vs function is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by authority vs function.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, authority vs function becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does authority vs function hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=authority vs function; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain authority vs function? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Verification Queue - Operating Card 01

Core proposition

Verification queue is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by verification queue.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, verification queue becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does verification queue hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=verification queue; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain verification queue? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Fact-Model Split - Operating Card 01

Core proposition

Fact-model split is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by fact-model split.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, fact-model split becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does fact-model split hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=fact-model split; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain fact-model split? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Scope Boundary - Operating Card 01

Core proposition

Scope boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scope boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scope boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scope boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scope boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scope boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Source Ecology - Operating Card 02

Core proposition

Source ecology is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source ecology.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source ecology becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source ecology hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source ecology; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source ecology? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Primary Source - Operating Card 02

Core proposition

Primary source is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by primary source.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, primary source becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does primary source hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=primary source; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain primary source? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Secondary Synthesis - Operating Card 02

Core proposition

Secondary synthesis is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by secondary synthesis.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, secondary synthesis becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does secondary synthesis hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=secondary synthesis; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain secondary synthesis? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Ledger - Operating Card 02

Core proposition

Claim ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Evidence Rank - Operating Card 02

Core proposition

Evidence rank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence rank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence rank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence rank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence rank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence rank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Citation Discipline - Operating Card 02

Core proposition

Citation discipline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by citation discipline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, citation discipline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does citation discipline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=citation discipline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain citation discipline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Uncertainty Map - Operating Card 02

Core proposition

Uncertainty map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by uncertainty map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, uncertainty map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does uncertainty map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=uncertainty map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain uncertainty map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Contradiction Table - Operating Card 02

Core proposition

Contradiction table is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contradiction table.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contradiction table becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contradiction table hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contradiction table; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contradiction table? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Research Question - Operating Card 02

Core proposition

Research question is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research question.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research question becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research question hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research question; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research question? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Mechanism Extraction - Operating Card 02

Core proposition

Mechanism extraction is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism extraction.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism extraction becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism extraction hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism extraction; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism extraction? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Source Triangulation - Operating Card 02

Core proposition

Source triangulation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source triangulation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source triangulation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source triangulation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source triangulation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source triangulation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Authority Vs Function - Operating Card 02

Core proposition

Authority vs function is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by authority vs function.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, authority vs function becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does authority vs function hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=authority vs function; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain authority vs function? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Verification Queue - Operating Card 02

Core proposition

Verification queue is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by verification queue.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, verification queue becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does verification queue hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=verification queue; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain verification queue? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Fact-Model Split - Operating Card 02

Core proposition

Fact-model split is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by fact-model split.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, fact-model split becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does fact-model split hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=fact-model split; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain fact-model split? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Scope Boundary - Operating Card 02

Core proposition

Scope boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scope boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scope boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scope boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scope boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scope boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Source Ecology - Operating Card 03

Core proposition

Source ecology is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source ecology.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source ecology becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source ecology hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source ecology; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source ecology? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Primary Source - Operating Card 03

Core proposition

Primary source is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by primary source.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, primary source becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does primary source hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=primary source; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain primary source? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Secondary Synthesis - Operating Card 03

Core proposition

Secondary synthesis is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by secondary synthesis.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, secondary synthesis becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does secondary synthesis hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=secondary synthesis; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain secondary synthesis? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Ledger - Operating Card 03

Core proposition

Claim ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Evidence Rank - Operating Card 03

Core proposition

Evidence rank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence rank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence rank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence rank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence rank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence rank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Citation Discipline - Operating Card 03

Core proposition

Citation discipline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by citation discipline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, citation discipline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does citation discipline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=citation discipline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain citation discipline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Uncertainty Map - Operating Card 03

Core proposition

Uncertainty map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by uncertainty map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, uncertainty map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does uncertainty map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=uncertainty map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain uncertainty map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Contradiction Table - Operating Card 03

Core proposition

Contradiction table is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contradiction table.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contradiction table becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contradiction table hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contradiction table; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contradiction table? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Research Question - Operating Card 03

Core proposition

Research question is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research question.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research question becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research question hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research question; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research question? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Mechanism Extraction - Operating Card 03

Core proposition

Mechanism extraction is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism extraction.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism extraction becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism extraction hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism extraction; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism extraction? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Source Triangulation - Operating Card 03

Core proposition

Source triangulation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source triangulation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source triangulation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source triangulation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source triangulation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source triangulation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Authority Vs Function - Operating Card 03

Core proposition

Authority vs function is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by authority vs function.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, authority vs function becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does authority vs function hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=authority vs function; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain authority vs function? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Verification Queue - Operating Card 03

Core proposition

Verification queue is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by verification queue.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, verification queue becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does verification queue hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=verification queue; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain verification queue? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Fact-Model Split - Operating Card 03

Core proposition

Fact-model split is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by fact-model split.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, fact-model split becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does fact-model split hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=fact-model split; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain fact-model split? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Scope Boundary - Operating Card 03

Core proposition

Scope boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scope boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scope boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scope boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scope boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scope boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

101 Clarity - Operating Card 01

Page 111 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

101 clarity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by 101 clarity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, 101 clarity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does 101 clarity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=101 clarity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain 101 clarity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reader Gradient - Operating Card 01

Core proposition

Reader gradient is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader gradient.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader gradient becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader gradient hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader gradient; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader gradient? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Cognitive Load - Operating Card 01

Core proposition

Cognitive load is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cognitive load.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cognitive load becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cognitive load hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cognitive load; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cognitive load? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Worked Example - Operating Card 01

Core proposition

Worked example is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by worked example.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, worked example becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does worked example hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=worked example; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain worked example? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Metaphor Control - Operating Card 01

Core proposition

Metaphor control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metaphor control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metaphor control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metaphor control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metaphor control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metaphor control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Definition Ladder - Operating Card 01

Core proposition

Definition ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by definition ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, definition ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does definition ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=definition ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain definition ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Beginner Bridge - Operating Card 01

Core proposition

Beginner bridge is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by beginner bridge.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, beginner bridge becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does beginner bridge hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=beginner bridge; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain beginner bridge? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Practice Block - Operating Card 01

Core proposition

Practice block is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by practice block.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, practice block becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does practice block hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=practice block; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain practice block? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Concept Check - Operating Card 01

Core proposition

Concept check is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept check.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept check becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept check hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept check; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept check? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Error Inoculation - Operating Card 01

Core proposition

Error inoculation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by error inoculation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, error inoculation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does error inoculation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=error inoculation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain error inoculation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Plain Language - Operating Card 01

Core proposition

Plain language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by plain language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, plain language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does plain language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=plain language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain plain language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Deep Simplicity - Operating Card 01

Core proposition

Deep simplicity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by deep simplicity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, deep simplicity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does deep simplicity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=deep simplicity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain deep simplicity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Scaffold Sequence - Operating Card 01

Core proposition

Scaffold sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scaffold sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scaffold sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scaffold sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scaffold sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scaffold sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Transfer Exercise - Operating Card 01

Core proposition

Transfer exercise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by transfer exercise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, transfer exercise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does transfer exercise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=transfer exercise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain transfer exercise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Schema Building - Operating Card 01

Core proposition

Schema building is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by schema building.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, schema building becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does schema building hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=schema building; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain schema building? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

101 Clarity - Operating Card 02

Page 126 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

101 clarity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by 101 clarity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, 101 clarity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does 101 clarity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=101 clarity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain 101 clarity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reader Gradient - Operating Card 02

Core proposition

Reader gradient is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader gradient.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader gradient becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader gradient hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader gradient; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader gradient? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Cognitive Load - Operating Card 02

Core proposition

Cognitive load is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cognitive load.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cognitive load becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cognitive load hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cognitive load; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cognitive load? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Worked Example - Operating Card 02

Core proposition

Worked example is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by worked example.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, worked example becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does worked example hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=worked example; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain worked example? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Metaphor Control - Operating Card 02

Core proposition

Metaphor control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metaphor control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metaphor control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metaphor control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metaphor control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metaphor control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Definition Ladder - Operating Card 02

Core proposition

Definition ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by definition ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, definition ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does definition ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=definition ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain definition ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Beginner Bridge - Operating Card 02

Page 132 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Beginner bridge is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by beginner bridge.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, beginner bridge becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does beginner bridge hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=beginner bridge; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain beginner bridge? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Practice Block - Operating Card 02

Core proposition

Practice block is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by practice block.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, practice block becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does practice block hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=practice block; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain practice block? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Concept Check - Operating Card 02

Core proposition

Concept check is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept check.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept check becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept check hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept check; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept check? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Error Inoculation - Operating Card 02

Core proposition

Error inoculation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by error inoculation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, error inoculation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does error inoculation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=error inoculation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain error inoculation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Plain Language - Operating Card 02

Core proposition

Plain language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by plain language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, plain language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does plain language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=plain language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain plain language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Deep Simplicity - Operating Card 02

Core proposition

Deep simplicity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by deep simplicity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, deep simplicity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does deep simplicity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=deep simplicity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain deep simplicity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Scaffold Sequence - Operating Card 02

Core proposition

Scaffold sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scaffold sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scaffold sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scaffold sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scaffold sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scaffold sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Transfer Exercise - Operating Card 02

Core proposition

Transfer exercise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by transfer exercise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, transfer exercise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does transfer exercise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=transfer exercise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain transfer exercise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Schema Building - Operating Card 02

Core proposition

Schema building is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by schema building.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, schema building becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does schema building hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=schema building; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain schema building? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

101 Clarity - Operating Card 03

Page 141 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

101 clarity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by 101 clarity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, 101 clarity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does 101 clarity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=101 clarity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain 101 clarity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reader Gradient - Operating Card 03

Core proposition

Reader gradient is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader gradient.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader gradient becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader gradient hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader gradient; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader gradient? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Cognitive Load - Operating Card 03

Core proposition

Cognitive load is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cognitive load.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cognitive load becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cognitive load hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cognitive load; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cognitive load? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Worked Example - Operating Card 03

Core proposition

Worked example is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by worked example.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, worked example becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does worked example hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=worked example; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain worked example? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Metaphor Control - Operating Card 03

Core proposition

Metaphor control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metaphor control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metaphor control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metaphor control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metaphor control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metaphor control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Definition Ladder - Operating Card 03

Core proposition

Definition ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by definition ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, definition ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does definition ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=definition ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain definition ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Beginner Bridge - Operating Card 03

Page 147 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Beginner bridge is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by beginner bridge.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, beginner bridge becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does beginner bridge hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=beginner bridge; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain beginner bridge? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Practice Block - Operating Card 03

Core proposition

Practice block is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by practice block.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, practice block becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does practice block hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=practice block; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain practice block? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Concept Check - Operating Card 03

Core proposition

Concept check is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept check.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept check becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept check hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept check; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept check? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Error Inoculation - Operating Card 03

Page 150 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Error inoculation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by error inoculation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, error inoculation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does error inoculation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=error inoculation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain error inoculation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Plain Language - Operating Card 03

Core proposition

Plain language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by plain language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, plain language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does plain language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=plain language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain plain language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Deep Simplicity - Operating Card 03

Core proposition

Deep simplicity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by deep simplicity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, deep simplicity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does deep simplicity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=deep simplicity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain deep simplicity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Scaffold Sequence - Operating Card 03

Core proposition

Scaffold sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by scaffold sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, scaffold sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does scaffold sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=scaffold sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain scaffold sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Transfer Exercise - Operating Card 03

Page 154 / 700 - P3 - Evidence rank: Synthesis + operational protocol - Agent: Graph Agent

Core proposition

Transfer exercise is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by transfer exercise.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, transfer exercise becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does transfer exercise hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=transfer exercise; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain transfer exercise? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Schema Building - Operating Card 03

Core proposition

Schema building is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by schema building.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, schema building becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does schema building hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=schema building; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain schema building? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Generative Kernel - Operating Card 01

Core proposition

Generative kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by generative kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, generative kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does generative kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=generative kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain generative kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Distinction Density - Operating Card 01

Core proposition

Distinction density is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by distinction density.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, distinction density becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does distinction density hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=distinction density; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain distinction density? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Mcd Writing - Operating Card 01

Core proposition

Mcd writing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD writing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD writing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD writing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD writing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD writing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Qmcd Model Map - Operating Card 01

Core proposition

Qmcd model map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QMCD model map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QMCD model map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QMCD model map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QMCD model map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QMCD model map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Chapter Compression - Operating Card 01

Page 160 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Chapter compression is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter compression.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter compression becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter compression hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter compression; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter compression? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Argument Spine - Operating Card 01

Core proposition

Argument spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Mechanism Paragraph - Operating Card 01

Core proposition

Mechanism paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Anti-Summary - Operating Card 01

Core proposition

Anti-summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by anti-summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, anti-summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does anti-summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=anti-summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain anti-summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Reader Transformation - Operating Card 01

Core proposition

Reader transformation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader transformation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader transformation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader transformation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader transformation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader transformation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Narrative Logic - Operating Card 01

Core proposition

Narrative logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by narrative logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, narrative logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does narrative logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=narrative logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain narrative logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Technical Paragraph - Operating Card 01

Page 166 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Technical paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Research Paragraph - Operating Card 01

Page 167 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Research paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Decision Paragraph - Operating Card 01

Core proposition

Decision paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by decision paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, decision paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does decision paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=decision paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain decision paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Concept Regeneration - Operating Card 01

Core proposition

Concept regeneration is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept regeneration.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept regeneration becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept regeneration hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept regeneration; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept regeneration? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Writing Rhythm - Operating Card 01

Core proposition

Writing rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by writing rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, writing rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does writing rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=writing rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain writing rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Generative Kernel - Operating Card 02

Core proposition

Generative kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by generative kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, generative kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does generative kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=generative kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain generative kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Distinction Density - Operating Card 02

Page 172 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Distinction density is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by distinction density.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, distinction density becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does distinction density hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=distinction density; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain distinction density? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Mcd Writing - Operating Card 02

Page 173 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Mcd writing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD writing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD writing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD writing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD writing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD writing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Qmcd Model Map - Operating Card 02

Core proposition

Qmcd model map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QMCD model map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QMCD model map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QMCD model map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QMCD model map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QMCD model map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Chapter Compression - Operating Card 02

Page 175 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Chapter compression is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter compression.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter compression becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter compression hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter compression; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter compression? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Argument Spine - Operating Card 02

Core proposition

Argument spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Mechanism Paragraph - Operating Card 02

Core proposition

Mechanism paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Anti-Summary - Operating Card 02

Page 178 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Anti-summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by anti-summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, anti-summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does anti-summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=anti-summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain anti-summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Reader Transformation - Operating Card 02

Core proposition

Reader transformation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader transformation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader transformation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader transformation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader transformation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader transformation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Narrative Logic - Operating Card 02

Core proposition

Narrative logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by narrative logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, narrative logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does narrative logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=narrative logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain narrative logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Technical Paragraph - Operating Card 02

Page 181 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Technical paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Research Paragraph - Operating Card 02

Page 182 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Research paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Decision Paragraph - Operating Card 02

Core proposition

Decision paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by decision paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, decision paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does decision paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=decision paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain decision paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Concept Regeneration - Operating Card 02

Core proposition

Concept regeneration is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept regeneration.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept regeneration becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept regeneration hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept regeneration; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept regeneration? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Writing Rhythm - Operating Card 02

Core proposition

Writing rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by writing rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, writing rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does writing rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=writing rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain writing rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Generative Kernel - Operating Card 03

Core proposition

Generative kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by generative kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, generative kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does generative kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=generative kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain generative kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Distinction Density - Operating Card 03

Core proposition

Distinction density is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by distinction density.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, distinction density becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does distinction density hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=distinction density; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain distinction density? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Mcd Writing - Operating Card 03

Core proposition

Mcd writing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD writing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD writing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD writing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD writing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD writing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Qmcd Model Map - Operating Card 03

Page 189 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Qmcd model map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QMCD model map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QMCD model map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QMCD model map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QMCD model map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QMCD model map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Chapter Compression - Operating Card 03

Page 190 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Chapter compression is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter compression.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter compression becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter compression hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter compression; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter compression? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Argument Spine - Operating Card 03

Core proposition

Argument spine is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument spine.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument spine becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument spine hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument spine; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument spine? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Mechanism Paragraph - Operating Card 03

Page 192 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Mechanism paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by mechanism paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, mechanism paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does mechanism paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=mechanism paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain mechanism paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Anti-Summary - Operating Card 03

Core proposition

Anti-summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by anti-summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, anti-summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does anti-summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=anti-summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain anti-summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Reader Transformation - Operating Card 03

Core proposition

Reader transformation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader transformation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader transformation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader transformation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader transformation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader transformation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Narrative Logic - Operating Card 03

Core proposition

Narrative logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by narrative logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, narrative logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does narrative logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=narrative logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain narrative logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Technical Paragraph - Operating Card 03

Core proposition

Technical paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Research Paragraph - Operating Card 03

Page 197 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Research paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Decision Paragraph - Operating Card 03

Core proposition

Decision paragraph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by decision paragraph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, decision paragraph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does decision paragraph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=decision paragraph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain decision paragraph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Concept Regeneration - Operating Card 03

Core proposition

Concept regeneration is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept regeneration.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept regeneration becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept regeneration hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept regeneration; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept regeneration? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Writing Rhythm - Operating Card 03

Page 200 / 700 - P4 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Writing rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by writing rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, writing rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does writing rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=writing rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain writing rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Visual Hierarchy - Operating Card 01

Core proposition

Visual hierarchy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by visual hierarchy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, visual hierarchy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does visual hierarchy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=visual hierarchy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain visual hierarchy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Grid System - Operating Card 01

Core proposition

Grid system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by grid system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, grid system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does grid system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=grid system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain grid system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Dense Page Grammar - Operating Card 01

Core proposition

Dense page grammar is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dense page grammar.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dense page grammar becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dense page grammar hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dense page grammar; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dense page grammar? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

White-Space Governance - Operating Card 01

Core proposition

White-space governance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by white-space governance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, white-space governance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does white-space governance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=white-space governance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain white-space governance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Callout Architecture - Operating Card 01

Page 205 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Callout architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by callout architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, callout architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does callout architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=callout architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain callout architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Table Economy - Operating Card 01

Page 206 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Skill Agent

Core proposition

Table economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by table economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, table economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does table economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=table economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain table economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Diagram Placeholder - Operating Card 01

Core proposition

Diagram placeholder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram placeholder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram placeholder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram placeholder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram placeholder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram placeholder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Thumbnail-First Cover - Operating Card 01

Core proposition

Thumbnail-first cover is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by thumbnail-first cover.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, thumbnail-first cover becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does thumbnail-first cover hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=thumbnail-first cover; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain thumbnail-first cover? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Neo-Abstract Aesthetic - Operating Card 01

Core proposition

Neo-abstract aesthetic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by neo-abstract aesthetic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, neo-abstract aesthetic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does neo-abstract aesthetic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=neo-abstract aesthetic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain neo-abstract aesthetic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Flat Design - Operating Card 01

Core proposition

Flat design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by flat design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, flat design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does flat design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=flat design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain flat design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Contrast Ladder - Operating Card 01

Core proposition

Contrast ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contrast ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contrast ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contrast ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contrast ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contrast ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Caption Logic - Operating Card 01

Core proposition

Caption logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by caption logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, caption logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does caption logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=caption logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain caption logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Information Blocks - Operating Card 01

Page 213 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Deployment Agent

Core proposition

Information blocks is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by information blocks.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, information blocks becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does information blocks hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=information blocks; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain information blocks? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Rhythm - Operating Card 01

Core proposition

Page rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Accessibility Design - Operating Card 01

Core proposition

Accessibility design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Visual Hierarchy - Operating Card 02

Core proposition

Visual hierarchy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by visual hierarchy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, visual hierarchy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does visual hierarchy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=visual hierarchy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain visual hierarchy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Grid System - Operating Card 02

Core proposition

Grid system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by grid system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, grid system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does grid system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=grid system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain grid system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Dense Page Grammar - Operating Card 02

Core proposition

Dense page grammar is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dense page grammar.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dense page grammar becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dense page grammar hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dense page grammar; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dense page grammar? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

White-Space Governance - Operating Card 02

Page 219 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Deployment Agent

Core proposition

White-space governance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by white-space governance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, white-space governance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does white-space governance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=white-space governance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain white-space governance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Callout Architecture - Operating Card 02

Core proposition

Callout architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by callout architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, callout architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does callout architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=callout architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain callout architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Table Economy - Operating Card 02

Page 221 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Skill Agent

Core proposition

Table economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by table economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, table economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does table economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=table economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain table economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Diagram Placeholder - Operating Card 02

Core proposition

Diagram placeholder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram placeholder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram placeholder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram placeholder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram placeholder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram placeholder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Thumbnail-First Cover - Operating Card 02

Core proposition

Thumbnail-first cover is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by thumbnail-first cover.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, thumbnail-first cover becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does thumbnail-first cover hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=thumbnail-first cover; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain thumbnail-first cover? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Neo-Abstract Aesthetic - Operating Card 02

Core proposition

Neo-abstract aesthetic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by neo-abstract aesthetic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, neo-abstract aesthetic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does neo-abstract aesthetic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=neo-abstract aesthetic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain neo-abstract aesthetic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Flat Design - Operating Card 02

Page 225 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Deployment Agent

Core proposition

Flat design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by flat design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, flat design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does flat design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=flat design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain flat design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Contrast Ladder - Operating Card 02

Page 226 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Contrast ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contrast ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contrast ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contrast ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contrast ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contrast ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Caption Logic - Operating Card 02

Page 227 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Skill Agent

Core proposition

Caption logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by caption logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, caption logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does caption logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=caption logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain caption logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Information Blocks - Operating Card 02

Core proposition

Information blocks is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by information blocks.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, information blocks becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does information blocks hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=information blocks; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain information blocks? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Rhythm - Operating Card 02

Core proposition

Page rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Accessibility Design - Operating Card 02

Core proposition

Accessibility design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Visual Hierarchy - Operating Card 03

Core proposition

Visual hierarchy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by visual hierarchy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, visual hierarchy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does visual hierarchy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=visual hierarchy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain visual hierarchy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Grid System - Operating Card 03

Core proposition

Grid system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by grid system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, grid system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does grid system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=grid system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain grid system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Dense Page Grammar - Operating Card 03

Core proposition

Dense page grammar is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dense page grammar.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dense page grammar becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dense page grammar hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dense page grammar; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dense page grammar? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

White-Space Governance - Operating Card 03

Core proposition

White-space governance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by white-space governance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, white-space governance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does white-space governance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=white-space governance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain white-space governance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Callout Architecture - Operating Card 03

Core proposition

Callout architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by callout architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, callout architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does callout architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=callout architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain callout architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Table Economy - Operating Card 03

Core proposition

Table economy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by table economy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, table economy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does table economy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=table economy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain table economy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Diagram Placeholder - Operating Card 03

Core proposition

Diagram placeholder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram placeholder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram placeholder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram placeholder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram placeholder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram placeholder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Thumbnail-First Cover - Operating Card 03

Core proposition

Thumbnail-first cover is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by thumbnail-first cover.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, thumbnail-first cover becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does thumbnail-first cover hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=thumbnail-first cover; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain thumbnail-first cover? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Neo-Abstract Aesthetic - Operating Card 03

Core proposition

Neo-abstract aesthetic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by neo-abstract aesthetic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, neo-abstract aesthetic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does neo-abstract aesthetic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=neo-abstract aesthetic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain neo-abstract aesthetic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Flat Design - Operating Card 03

Page 240 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Deployment Agent

Core proposition

Flat design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by flat design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, flat design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does flat design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=flat design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain flat design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Contrast Ladder - Operating Card 03

Core proposition

Contrast ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by contrast ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, contrast ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does contrast ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=contrast ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain contrast ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Caption Logic - Operating Card 03

Core proposition

Caption logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by caption logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, caption logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does caption logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=caption logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain caption logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Information Blocks - Operating Card 03

Page 243 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Deployment Agent

Core proposition

Information blocks is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by information blocks.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, information blocks becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does information blocks hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=information blocks; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain information blocks? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Page Rhythm - Operating Card 03

Page 244 / 700 - P5 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Page rhythm is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by page rhythm.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, page rhythm becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does page rhythm hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=page rhythm; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain page rhythm? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Accessibility Design - Operating Card 03

Core proposition

Accessibility design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Preserve Strong Material - Operating Card 01

Core proposition

Preserve strong material is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by preserve strong material.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, preserve strong material becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does preserve strong material hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=preserve strong material; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain preserve strong material? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Detect Weak Page - Operating Card 01

Core proposition

Detect weak page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by detect weak page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, detect weak page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does detect weak page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=detect weak page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain detect weak page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Rewrite Ladder - Operating Card 01

Core proposition

Rewrite ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rewrite ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rewrite ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rewrite ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rewrite ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rewrite ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Style Transfer Boundary - Operating Card 01

Page 249 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Retrieval Agent

Core proposition

Style transfer boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by style transfer boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, style transfer boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does style transfer boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=style transfer boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain style transfer boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Source Fidelity - Operating Card 01

Page 250 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Source fidelity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source fidelity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source fidelity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source fidelity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source fidelity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source fidelity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Chapter Repair - Operating Card 01

Page 251 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Evidence Agent

Core proposition

Chapter repair is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter repair.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter repair becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter repair hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter repair; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter repair? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Argument Tightening - Operating Card 01

Core proposition

Argument tightening is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument tightening.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument tightening becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument tightening hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument tightening; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument tightening? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Example Injection - Operating Card 01

Core proposition

Example injection is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by example injection.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, example injection becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does example injection hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=example injection; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain example injection? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Redundancy Removal - Operating Card 01

Core proposition

Redundancy removal is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by redundancy removal.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, redundancy removal becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does redundancy removal hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=redundancy removal; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain redundancy removal? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Claim Upgrade - Operating Card 01

Core proposition

Claim upgrade is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim upgrade.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim upgrade becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim upgrade hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim upgrade; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim upgrade? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Reader Test - Operating Card 01

Core proposition

Reader test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Before-After Matrix - Operating Card 01

Core proposition

Before-after matrix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by before-after matrix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, before-after matrix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does before-after matrix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=before-after matrix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain before-after matrix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Copyright Caution - Operating Card 01

Core proposition

Copyright caution is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright caution.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright caution becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright caution hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright caution; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright caution? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Voice Preservation - Operating Card 01

Core proposition

Voice preservation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by voice preservation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, voice preservation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does voice preservation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=voice preservation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain voice preservation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Repair Log - Operating Card 01

Core proposition

Repair log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Preserve Strong Material - Operating Card 02

Core proposition

Preserve strong material is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by preserve strong material.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, preserve strong material becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does preserve strong material hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=preserve strong material; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain preserve strong material? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Detect Weak Page - Operating Card 02

Core proposition

Detect weak page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by detect weak page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, detect weak page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does detect weak page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=detect weak page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain detect weak page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Rewrite Ladder - Operating Card 02

Core proposition

Rewrite ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rewrite ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rewrite ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rewrite ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rewrite ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rewrite ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Style Transfer Boundary - Operating Card 02

Page 264 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Retrieval Agent

Core proposition

Style transfer boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by style transfer boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, style transfer boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does style transfer boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=style transfer boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain style transfer boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Source Fidelity - Operating Card 02

Core proposition

Source fidelity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source fidelity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source fidelity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source fidelity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source fidelity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source fidelity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Chapter Repair - Operating Card 02

Core proposition

Chapter repair is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter repair.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter repair becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter repair hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter repair; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter repair? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Argument Tightening - Operating Card 02

Core proposition

Argument tightening is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument tightening.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument tightening becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument tightening hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument tightening; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument tightening? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Example Injection - Operating Card 02

Core proposition

Example injection is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by example injection.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, example injection becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does example injection hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=example injection; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain example injection? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Redundancy Removal - Operating Card 02

Core proposition

Redundancy removal is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by redundancy removal.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, redundancy removal becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does redundancy removal hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=redundancy removal; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain redundancy removal? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Claim Upgrade - Operating Card 02

Core proposition

Claim upgrade is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim upgrade.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim upgrade becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim upgrade hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim upgrade; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim upgrade? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Reader Test - Operating Card 02

Core proposition

Reader test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Before-After Matrix - Operating Card 02

Core proposition

Before-after matrix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by before-after matrix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, before-after matrix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does before-after matrix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=before-after matrix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain before-after matrix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Copyright Caution - Operating Card 02

Core proposition

Copyright caution is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright caution.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright caution becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright caution hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright caution; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright caution? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Voice Preservation - Operating Card 02

Core proposition

Voice preservation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by voice preservation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, voice preservation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does voice preservation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=voice preservation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain voice preservation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Repair Log - Operating Card 02

Core proposition

Repair log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Preserve Strong Material - Operating Card 03

Core proposition

Preserve strong material is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by preserve strong material.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, preserve strong material becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does preserve strong material hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=preserve strong material; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain preserve strong material? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Detect Weak Page - Operating Card 03

Core proposition

Detect weak page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by detect weak page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, detect weak page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does detect weak page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=detect weak page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain detect weak page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Rewrite Ladder - Operating Card 03

Core proposition

Rewrite ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rewrite ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rewrite ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rewrite ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rewrite ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rewrite ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Style Transfer Boundary - Operating Card 03

Page 279 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Retrieval Agent

Core proposition

Style transfer boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by style transfer boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, style transfer boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does style transfer boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=style transfer boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain style transfer boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Source Fidelity - Operating Card 03

Page 280 / 700 - P6 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Source fidelity is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source fidelity.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source fidelity becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source fidelity hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source fidelity; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source fidelity? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Chapter Repair - Operating Card 03

Core proposition

Chapter repair is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chapter repair.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chapter repair becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chapter repair hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chapter repair; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chapter repair? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Argument Tightening - Operating Card 03

Core proposition

Argument tightening is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by argument tightening.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, argument tightening becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does argument tightening hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=argument tightening; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain argument tightening? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Example Injection - Operating Card 03

Core proposition

Example injection is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by example injection.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, example injection becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does example injection hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=example injection; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain example injection? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Redundancy Removal - Operating Card 03

Core proposition

Redundancy removal is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by redundancy removal.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, redundancy removal becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does redundancy removal hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=redundancy removal; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain redundancy removal? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Claim Upgrade - Operating Card 03

Core proposition

Claim upgrade is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim upgrade.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim upgrade becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim upgrade hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim upgrade; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim upgrade? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Reader Test - Operating Card 03

Core proposition

Reader test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Before-After Matrix - Operating Card 03

Core proposition

Before-after matrix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by before-after matrix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, before-after matrix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does before-after matrix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=before-after matrix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain before-after matrix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Copyright Caution - Operating Card 03

Core proposition

Copyright caution is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright caution.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright caution becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright caution hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright caution; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright caution? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Voice Preservation - Operating Card 03

Core proposition

Voice preservation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by voice preservation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, voice preservation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does voice preservation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=voice preservation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain voice preservation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Repair Log - Operating Card 03

Core proposition

Repair log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

System Architecture - Operating Card 01

Core proposition

System architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by system architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, system architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does system architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=system architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain system architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Workflow Tutorial - Operating Card 01

Core proposition

Workflow tutorial is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by workflow tutorial.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, workflow tutorial becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does workflow tutorial hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=workflow tutorial; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain workflow tutorial? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Stack Map - Operating Card 01

Core proposition

Stack map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stack map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stack map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stack map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stack map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stack map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Tool Comparison - Operating Card 01

Core proposition

Tool comparison is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool comparison.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool comparison becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool comparison hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool comparison; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool comparison? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Api Explanation - Operating Card 01

Core proposition

Api explanation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by API explanation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, API explanation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does API explanation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=API explanation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain API explanation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Cloud Pattern - Operating Card 01

Core proposition

Cloud pattern is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cloud pattern.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cloud pattern becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cloud pattern hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cloud pattern; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cloud pattern? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Ai Agents - Operating Card 01

Core proposition

Ai agents is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI agents.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI agents becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI agents hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI agents; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI agents? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Devops Flow - Operating Card 01

Core proposition

Devops flow is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by DevOps flow.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, DevOps flow becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does DevOps flow hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=DevOps flow; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain DevOps flow? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Cyber Risk - Operating Card 01

Core proposition

Cyber risk is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cyber risk.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cyber risk becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cyber risk hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cyber risk; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cyber risk? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Data Pipeline - Operating Card 01

Core proposition

Data pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by data pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, data pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does data pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=data pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain data pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Diagram-First Teaching - Operating Card 01

Core proposition

Diagram-first teaching is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram-first teaching.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram-first teaching becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram-first teaching hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram-first teaching; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram-first teaching? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Technical Glossary - Operating Card 01

Core proposition

Technical glossary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical glossary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical glossary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical glossary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical glossary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical glossary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Implementation Lab - Operating Card 01

Core proposition

Implementation lab is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by implementation lab.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, implementation lab becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does implementation lab hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=implementation lab; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain implementation lab? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Debug Appendix - Operating Card 01

Core proposition

Debug appendix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by debug appendix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, debug appendix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does debug appendix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=debug appendix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain debug appendix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Benchmark Page - Operating Card 01

Page 305 / 700 - P7 - Evidence rank: Synthesis + operational protocol - Agent: Versioning Agent

Core proposition

Benchmark page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

System Architecture - Operating Card 02

Core proposition

System architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by system architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, system architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does system architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=system architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain system architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Workflow Tutorial - Operating Card 02

Core proposition

Workflow tutorial is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by workflow tutorial.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, workflow tutorial becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does workflow tutorial hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=workflow tutorial; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain workflow tutorial? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Stack Map - Operating Card 02

Core proposition

Stack map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stack map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stack map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stack map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stack map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stack map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Tool Comparison - Operating Card 02

Core proposition

Tool comparison is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool comparison.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool comparison becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool comparison hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool comparison; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool comparison? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Api Explanation - Operating Card 02

Core proposition

Api explanation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by API explanation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, API explanation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does API explanation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=API explanation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain API explanation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Cloud Pattern - Operating Card 02

Core proposition

Cloud pattern is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cloud pattern.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cloud pattern becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cloud pattern hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cloud pattern; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cloud pattern? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Ai Agents - Operating Card 02

Core proposition

Ai agents is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI agents.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI agents becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI agents hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI agents; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI agents? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Devops Flow - Operating Card 02

Core proposition

Devops flow is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by DevOps flow.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, DevOps flow becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does DevOps flow hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=DevOps flow; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain DevOps flow? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Cyber Risk - Operating Card 02

Core proposition

Cyber risk is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cyber risk.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cyber risk becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cyber risk hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cyber risk; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cyber risk? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Data Pipeline - Operating Card 02

Core proposition

Data pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by data pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, data pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does data pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=data pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain data pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Diagram-First Teaching - Operating Card 02

Core proposition

Diagram-first teaching is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram-first teaching.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram-first teaching becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram-first teaching hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram-first teaching; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram-first teaching? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Technical Glossary - Operating Card 02

Core proposition

Technical glossary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical glossary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical glossary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical glossary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical glossary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical glossary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Implementation Lab - Operating Card 02

Core proposition

Implementation lab is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by implementation lab.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, implementation lab becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does implementation lab hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=implementation lab; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain implementation lab? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Debug Appendix - Operating Card 02

Core proposition

Debug appendix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by debug appendix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, debug appendix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does debug appendix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=debug appendix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain debug appendix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Benchmark Page - Operating Card 02

Core proposition

Benchmark page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

System Architecture - Operating Card 03

Core proposition

System architecture is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by system architecture.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, system architecture becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does system architecture hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=system architecture; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain system architecture? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Workflow Tutorial - Operating Card 03

Core proposition

Workflow tutorial is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by workflow tutorial.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, workflow tutorial becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does workflow tutorial hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=workflow tutorial; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain workflow tutorial? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Stack Map - Operating Card 03

Core proposition

Stack map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stack map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stack map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stack map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stack map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stack map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Tool Comparison - Operating Card 03

Core proposition

Tool comparison is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool comparison.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool comparison becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool comparison hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool comparison; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool comparison? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Api Explanation - Operating Card 03

Core proposition

Api explanation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by API explanation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, API explanation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does API explanation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=API explanation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain API explanation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Cloud Pattern - Operating Card 03

Core proposition

Cloud pattern is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cloud pattern.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cloud pattern becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cloud pattern hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cloud pattern; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cloud pattern? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Ai Agents - Operating Card 03

Core proposition

Ai agents is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI agents.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI agents becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI agents hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI agents; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI agents? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Devops Flow - Operating Card 03

Core proposition

Devops flow is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by DevOps flow.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, DevOps flow becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does DevOps flow hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=DevOps flow; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain DevOps flow? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Cyber Risk - Operating Card 03

Core proposition

Cyber risk is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cyber risk.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cyber risk becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cyber risk hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cyber risk; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cyber risk? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Data Pipeline - Operating Card 03

Core proposition

Data pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by data pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, data pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does data pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=data pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain data pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Diagram-First Teaching - Operating Card 03

Core proposition

Diagram-first teaching is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by diagram-first teaching.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, diagram-first teaching becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does diagram-first teaching hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=diagram-first teaching; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain diagram-first teaching? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Technical Glossary - Operating Card 03

Core proposition

Technical glossary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by technical glossary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, technical glossary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does technical glossary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=technical glossary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain technical glossary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Implementation Lab - Operating Card 03

Core proposition

Implementation lab is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by implementation lab.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, implementation lab becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does implementation lab hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=implementation lab; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain implementation lab? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Debug Appendix - Operating Card 03

Core proposition

Debug appendix is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by debug appendix.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, debug appendix becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does debug appendix hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=debug appendix; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain debug appendix? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Benchmark Page - Operating Card 03

Page 335 / 700 - P7 - Evidence rank: Synthesis + operational protocol - Agent: Versioning Agent

Core proposition

Benchmark page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Psychoeducation - Operating Card 01

Core proposition

Psychoeducation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by psychoeducation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, psychoeducation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does psychoeducation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=psychoeducation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain psychoeducation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Safety Disclaimer - Operating Card 01

Core proposition

Safety disclaimer is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by safety disclaimer.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, safety disclaimer becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does safety disclaimer hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=safety disclaimer; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain safety disclaimer? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Symptom Boundary - Operating Card 01

Core proposition

Symptom boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by symptom boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, symptom boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does symptom boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=symptom boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain symptom boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Exercise Protocol - Operating Card 01

Core proposition

Exercise protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by exercise protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, exercise protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does exercise protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=exercise protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain exercise protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Habit Loop - Operating Card 01

Core proposition

Habit loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by habit loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, habit loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does habit loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=habit loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain habit loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Sleep System - Operating Card 01

Core proposition

Sleep system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by sleep system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, sleep system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does sleep system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=sleep system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain sleep system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Stress Map - Operating Card 01

Core proposition

Stress map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stress map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stress map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stress map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stress map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stress map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Cbt Worksheet - Operating Card 01

Core proposition

Cbt worksheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by CBT worksheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, CBT worksheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does CBT worksheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=CBT worksheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain CBT worksheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Somatic Practice - Operating Card 01

Core proposition

Somatic practice is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by somatic practice.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, somatic practice becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does somatic practice hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=somatic practice; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain somatic practice? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Risk Escalation - Operating Card 01

Core proposition

Risk escalation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by risk escalation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, risk escalation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does risk escalation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=risk escalation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain risk escalation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Evidence Humility - Operating Card 01

Core proposition

Evidence humility is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence humility.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence humility becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence humility hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence humility; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence humility? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Autonomy - Operating Card 01

Core proposition

Reader autonomy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader autonomy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader autonomy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader autonomy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader autonomy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader autonomy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Non-Diagnostic Language - Operating Card 01

Core proposition

Non-diagnostic language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by non-diagnostic language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, non-diagnostic language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does non-diagnostic language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=non-diagnostic language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain non-diagnostic language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Tracking Sheet - Operating Card 01

Core proposition

Tracking sheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tracking sheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tracking sheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tracking sheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tracking sheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tracking sheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Clinician Boundary - Operating Card 01

Page 350 / 700 - P8 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Clinician boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by clinician boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, clinician boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does clinician boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=clinician boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain clinician boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Psychoeducation - Operating Card 02

Core proposition

Psychoeducation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by psychoeducation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, psychoeducation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does psychoeducation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=psychoeducation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain psychoeducation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Safety Disclaimer - Operating Card 02

Core proposition

Safety disclaimer is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by safety disclaimer.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, safety disclaimer becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does safety disclaimer hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=safety disclaimer; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain safety disclaimer? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Symptom Boundary - Operating Card 02

Core proposition

Symptom boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by symptom boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, symptom boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does symptom boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=symptom boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain symptom boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Exercise Protocol - Operating Card 02

Core proposition

Exercise protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by exercise protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, exercise protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does exercise protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=exercise protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain exercise protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Habit Loop - Operating Card 02

Core proposition

Habit loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by habit loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, habit loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does habit loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=habit loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain habit loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Sleep System - Operating Card 02

Core proposition

Sleep system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by sleep system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, sleep system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does sleep system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=sleep system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain sleep system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Stress Map - Operating Card 02

Core proposition

Stress map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stress map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stress map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stress map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stress map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stress map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Cbt Worksheet - Operating Card 02

Core proposition

Cbt worksheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by CBT worksheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, CBT worksheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does CBT worksheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=CBT worksheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain CBT worksheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Somatic Practice - Operating Card 02

Core proposition

Somatic practice is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by somatic practice.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, somatic practice becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does somatic practice hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=somatic practice; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain somatic practice? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Risk Escalation - Operating Card 02

Core proposition

Risk escalation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by risk escalation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, risk escalation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does risk escalation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=risk escalation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain risk escalation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Evidence Humility - Operating Card 02

Core proposition

Evidence humility is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence humility.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence humility becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence humility hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence humility; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence humility? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Autonomy - Operating Card 02

Core proposition

Reader autonomy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader autonomy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader autonomy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader autonomy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader autonomy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader autonomy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Non-Diagnostic Language - Operating Card 02

Page 363 / 700 - P8 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Non-diagnostic language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by non-diagnostic language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, non-diagnostic language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does non-diagnostic language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=non-diagnostic language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain non-diagnostic language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Tracking Sheet - Operating Card 02

Core proposition

Tracking sheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tracking sheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tracking sheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tracking sheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tracking sheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tracking sheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Clinician Boundary - Operating Card 02

Core proposition

Clinician boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by clinician boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, clinician boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does clinician boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=clinician boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain clinician boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Psychoeducation - Operating Card 03

Core proposition

Psychoeducation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by psychoeducation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, psychoeducation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does psychoeducation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=psychoeducation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain psychoeducation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Safety Disclaimer - Operating Card 03

Core proposition

Safety disclaimer is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by safety disclaimer.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, safety disclaimer becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does safety disclaimer hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=safety disclaimer; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain safety disclaimer? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Symptom Boundary - Operating Card 03

Core proposition

Symptom boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by symptom boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, symptom boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does symptom boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=symptom boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain symptom boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Exercise Protocol - Operating Card 03

Core proposition

Exercise protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by exercise protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, exercise protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does exercise protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=exercise protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain exercise protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Habit Loop - Operating Card 03

Core proposition

Habit loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by habit loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, habit loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does habit loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=habit loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain habit loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Sleep System - Operating Card 03

Core proposition

Sleep system is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by sleep system.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, sleep system becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does sleep system hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=sleep system; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain sleep system? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Stress Map - Operating Card 03

Core proposition

Stress map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by stress map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, stress map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does stress map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=stress map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain stress map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Cbt Worksheet - Operating Card 03

Core proposition

Cbt worksheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by CBT worksheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, CBT worksheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does CBT worksheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=CBT worksheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain CBT worksheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Somatic Practice - Operating Card 03

Core proposition

Somatic practice is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by somatic practice.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, somatic practice becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does somatic practice hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=somatic practice; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain somatic practice? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Risk Escalation - Operating Card 03

Core proposition

Risk escalation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by risk escalation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, risk escalation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does risk escalation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=risk escalation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain risk escalation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Evidence Humility - Operating Card 03

Core proposition

Evidence humility is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence humility.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence humility becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence humility hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence humility; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence humility? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Autonomy - Operating Card 03

Core proposition

Reader autonomy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader autonomy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader autonomy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader autonomy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader autonomy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader autonomy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Non-Diagnostic Language - Operating Card 03

Page 378 / 700 - P8 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Non-diagnostic language is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by non-diagnostic language.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, non-diagnostic language becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does non-diagnostic language hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=non-diagnostic language; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain non-diagnostic language? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Tracking Sheet - Operating Card 03

Page 379 / 700 - P8 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Tracking sheet is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tracking sheet.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tracking sheet becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tracking sheet hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tracking sheet; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tracking sheet? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Clinician Boundary - Operating Card 03

Core proposition

Clinician boundary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by clinician boundary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, clinician boundary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does clinician boundary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=clinician boundary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain clinician boundary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Market Intent - Operating Card 01

Core proposition

Market intent is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market intent.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market intent becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market intent hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market intent; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market intent? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Design - Operating Card 01

Core proposition

Offer design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Pain-To-Mechanism - Operating Card 01

Core proposition

Pain-to-mechanism is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pain-to-mechanism.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pain-to-mechanism becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pain-to-mechanism hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pain-to-mechanism; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pain-to-mechanism? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Copy Ladder - Operating Card 01

Core proposition

Copy ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copy ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copy ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copy ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copy ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copy ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Landing Page - Operating Card 01

Core proposition

Landing page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by landing page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, landing page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does landing page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=landing page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain landing page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Hook Bank - Operating Card 01

Core proposition

Hook bank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hook bank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hook bank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hook bank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hook bank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hook bank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Case Study - Operating Card 01

Core proposition

Case study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by case study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, case study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does case study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=case study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain case study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Email Sequence - Operating Card 01

Core proposition

Email sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by email sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, email sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does email sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=email sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain email sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ad Angle - Operating Card 01

Core proposition

Ad angle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ad angle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ad angle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ad angle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ad angle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ad angle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Seo Cluster - Operating Card 01

Core proposition

Seo cluster is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO cluster.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO cluster becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO cluster hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO cluster; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO cluster? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Conversion Metric - Operating Card 01

Core proposition

Conversion metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by conversion metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, conversion metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does conversion metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=conversion metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain conversion metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Trust Proof - Operating Card 01

Core proposition

Trust proof is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust proof.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust proof becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust proof hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust proof; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust proof? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Objection Handling - Operating Card 01

Core proposition

Objection handling is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by objection handling.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, objection handling becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does objection handling hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=objection handling; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain objection handling? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ethical Urgency - Operating Card 01

Core proposition

Ethical urgency is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ethical urgency.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ethical urgency becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ethical urgency hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ethical urgency; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ethical urgency? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Positioning Map - Operating Card 01

Core proposition

Positioning map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by positioning map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, positioning map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does positioning map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=positioning map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain positioning map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Market Intent - Operating Card 02

Core proposition

Market intent is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market intent.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market intent becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market intent hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market intent; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market intent? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Design - Operating Card 02

Core proposition

Offer design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Pain-To-Mechanism - Operating Card 02

Core proposition

Pain-to-mechanism is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pain-to-mechanism.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pain-to-mechanism becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pain-to-mechanism hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pain-to-mechanism; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pain-to-mechanism? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Copy Ladder - Operating Card 02

Core proposition

Copy ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copy ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copy ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copy ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copy ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copy ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Landing Page - Operating Card 02

Page 400 / 700 - P9 - Evidence rank: Synthesis + operational protocol - Agent: Concept Agent

Core proposition

Landing page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by landing page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, landing page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does landing page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=landing page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain landing page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Hook Bank - Operating Card 02

Core proposition

Hook bank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hook bank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hook bank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hook bank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hook bank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hook bank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Case Study - Operating Card 02

Core proposition

Case study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by case study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, case study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does case study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=case study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain case study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Email Sequence - Operating Card 02

Core proposition

Email sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by email sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, email sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does email sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=email sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain email sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ad Angle - Operating Card 02

Core proposition

Ad angle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ad angle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ad angle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ad angle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ad angle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ad angle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Seo Cluster - Operating Card 02

Core proposition

Seo cluster is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO cluster.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO cluster becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO cluster hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO cluster; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO cluster? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Conversion Metric - Operating Card 02

Core proposition

Conversion metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by conversion metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, conversion metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does conversion metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=conversion metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain conversion metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Trust Proof - Operating Card 02

Core proposition

Trust proof is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust proof.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust proof becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust proof hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust proof; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust proof? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Objection Handling - Operating Card 02

Core proposition

Objection handling is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by objection handling.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, objection handling becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does objection handling hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=objection handling; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain objection handling? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ethical Urgency - Operating Card 02

Core proposition

Ethical urgency is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ethical urgency.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ethical urgency becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ethical urgency hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ethical urgency; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ethical urgency? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Positioning Map - Operating Card 02

Core proposition

Positioning map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by positioning map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, positioning map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does positioning map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=positioning map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain positioning map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Market Intent - Operating Card 03

Core proposition

Market intent is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market intent.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market intent becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market intent hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market intent; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market intent? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Offer Design - Operating Card 03

Core proposition

Offer design is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by offer design.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, offer design becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does offer design hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=offer design; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain offer design? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Pain-To-Mechanism - Operating Card 03

Core proposition

Pain-to-mechanism is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by pain-to-mechanism.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, pain-to-mechanism becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does pain-to-mechanism hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=pain-to-mechanism; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain pain-to-mechanism? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Copy Ladder - Operating Card 03

Core proposition

Copy ladder is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copy ladder.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copy ladder becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copy ladder hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copy ladder; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copy ladder? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Landing Page - Operating Card 03

Core proposition

Landing page is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by landing page.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, landing page becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does landing page hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=landing page; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain landing page? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Hook Bank - Operating Card 03

Core proposition

Hook bank is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hook bank.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hook bank becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hook bank hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hook bank; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hook bank? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Case Study - Operating Card 03

Core proposition

Case study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by case study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, case study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does case study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=case study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain case study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Email Sequence - Operating Card 03

Core proposition

Email sequence is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by email sequence.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, email sequence becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does email sequence hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=email sequence; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain email sequence? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ad Angle - Operating Card 03

Core proposition

Ad angle is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ad angle.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ad angle becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ad angle hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ad angle; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ad angle? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Seo Cluster - Operating Card 03

Core proposition

Seo cluster is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO cluster.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO cluster becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO cluster hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO cluster; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO cluster? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Conversion Metric - Operating Card 03

Core proposition

Conversion metric is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by conversion metric.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, conversion metric becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does conversion metric hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=conversion metric; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain conversion metric? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Trust Proof - Operating Card 03

Core proposition

Trust proof is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by trust proof.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, trust proof becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does trust proof hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=trust proof; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain trust proof? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Objection Handling - Operating Card 03

Core proposition

Objection handling is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by objection handling.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, objection handling becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does objection handling hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=objection handling; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain objection handling? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Ethical Urgency - Operating Card 03

Core proposition

Ethical urgency is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by ethical urgency.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, ethical urgency becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does ethical urgency hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=ethical urgency; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain ethical urgency? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Positioning Map - Operating Card 03

Core proposition

Positioning map is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by positioning map.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, positioning map becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does positioning map hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=positioning map; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain positioning map? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Metadata Pack - Operating Card 01

Core proposition

Metadata pack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metadata pack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metadata pack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metadata pack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metadata pack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metadata pack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Kdp Checklist - Operating Card 01

Core proposition

Kdp checklist is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by KDP checklist.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, KDP checklist becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does KDP checklist hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=KDP checklist; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain KDP checklist? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Ai Disclosure - Operating Card 01

Core proposition

Ai disclosure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI disclosure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI disclosure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI disclosure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI disclosure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI disclosure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Cover Specs - Operating Card 01

Core proposition

Cover specs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cover specs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cover specs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cover specs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cover specs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cover specs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Epub Export - Operating Card 01

Page 430 / 700 - P10 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Epub export is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by EPUB export.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, EPUB export becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does EPUB export hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=EPUB export; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain EPUB export? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Pdf Qa - Operating Card 01

Core proposition

Pdf qa is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by PDF QA.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, PDF QA becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does PDF QA hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=PDF QA; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain PDF QA? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Gumroad Listing - Operating Card 01

Core proposition

Gumroad listing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Gumroad listing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Gumroad listing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Gumroad listing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=Gumroad listing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Gumroad listing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Refund Policy - Operating Card 01

Core proposition

Refund policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by refund policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, refund policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does refund policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=refund policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain refund policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Seo Title - Operating Card 01

Core proposition

Seo title is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO title.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO title becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO title hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO title; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO title? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Keywords - Operating Card 01

Core proposition

Keywords is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by keywords.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, keywords becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does keywords hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=keywords; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain keywords? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Category Logic - Operating Card 01

Core proposition

Category logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by category logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, category logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does category logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=category logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain category logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Accessibility Metadata - Operating Card 01

Core proposition

Accessibility metadata is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility metadata.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility metadata becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility metadata hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility metadata; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility metadata? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Launch Plan - Operating Card 01

Core proposition

Launch plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by launch plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, launch plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does launch plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=launch plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain launch plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Analytics Loop - Operating Card 01

Core proposition

Analytics loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by analytics loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, analytics loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does analytics loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=analytics loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain analytics loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Platform Verification - Operating Card 01

Core proposition

Platform verification is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by platform verification.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, platform verification becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does platform verification hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=platform verification; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain platform verification? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Metadata Pack - Operating Card 02

Core proposition

Metadata pack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metadata pack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metadata pack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metadata pack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metadata pack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metadata pack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Kdp Checklist - Operating Card 02

Core proposition

Kdp checklist is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by KDP checklist.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, KDP checklist becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does KDP checklist hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=KDP checklist; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain KDP checklist? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Ai Disclosure - Operating Card 02

Core proposition

Ai disclosure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI disclosure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI disclosure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI disclosure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI disclosure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI disclosure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Cover Specs - Operating Card 02

Core proposition

Cover specs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cover specs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cover specs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cover specs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cover specs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cover specs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Epub Export - Operating Card 02

Core proposition

Epub export is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by EPUB export.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, EPUB export becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does EPUB export hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=EPUB export; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain EPUB export? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Pdf Qa - Operating Card 02

Core proposition

Pdf qa is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by PDF QA.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, PDF QA becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does PDF QA hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=PDF QA; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain PDF QA? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Gumroad Listing - Operating Card 02

Core proposition

Gumroad listing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Gumroad listing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Gumroad listing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Gumroad listing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=Gumroad listing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Gumroad listing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Refund Policy - Operating Card 02

Core proposition

Refund policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by refund policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, refund policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does refund policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=refund policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain refund policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Seo Title - Operating Card 02

Core proposition

Seo title is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO title.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO title becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO title hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO title; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO title? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Keywords - Operating Card 02

Core proposition

Keywords is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by keywords.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, keywords becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does keywords hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=keywords; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain keywords? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Category Logic - Operating Card 02

Core proposition

Category logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by category logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, category logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does category logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=category logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain category logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Accessibility Metadata - Operating Card 02

Core proposition

Accessibility metadata is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility metadata.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility metadata becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility metadata hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility metadata; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility metadata? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Launch Plan - Operating Card 02

Core proposition

Launch plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by launch plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, launch plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does launch plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=launch plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain launch plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Analytics Loop - Operating Card 02

Core proposition

Analytics loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by analytics loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, analytics loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does analytics loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=analytics loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain analytics loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Platform Verification - Operating Card 02

Core proposition

Platform verification is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by platform verification.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, platform verification becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does platform verification hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=platform verification; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain platform verification? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Metadata Pack - Operating Card 03

Core proposition

Metadata pack is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by metadata pack.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, metadata pack becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does metadata pack hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=metadata pack; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain metadata pack? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Kdp Checklist - Operating Card 03

Core proposition

Kdp checklist is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by KDP checklist.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, KDP checklist becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does KDP checklist hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=KDP checklist; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain KDP checklist? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Ai Disclosure - Operating Card 03

Core proposition

Ai disclosure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by AI disclosure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, AI disclosure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does AI disclosure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=AI disclosure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain AI disclosure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Cover Specs - Operating Card 03

Core proposition

Cover specs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by cover specs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, cover specs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does cover specs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=cover specs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain cover specs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Epub Export - Operating Card 03

Core proposition

Epub export is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by EPUB export.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, EPUB export becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does EPUB export hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=EPUB export; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain EPUB export? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Pdf Qa - Operating Card 03

Core proposition

Pdf qa is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by PDF QA.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, PDF QA becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does PDF QA hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=PDF QA; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain PDF QA? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Gumroad Listing - Operating Card 03

Core proposition

Gumroad listing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Gumroad listing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Gumroad listing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Gumroad listing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=Gumroad listing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Gumroad listing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Refund Policy - Operating Card 03

Page 463 / 700 - P10 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Refund policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by refund policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, refund policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does refund policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=refund policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain refund policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Seo Title - Operating Card 03

Core proposition

Seo title is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO title.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO title becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO title hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO title; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO title? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Keywords - Operating Card 03

Core proposition

Keywords is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by keywords.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, keywords becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does keywords hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=keywords; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain keywords? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Category Logic - Operating Card 03

Core proposition

Category logic is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by category logic.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, category logic becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does category logic hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=category logic; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain category logic? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Accessibility Metadata - Operating Card 03

Core proposition

Accessibility metadata is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by accessibility metadata.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, accessibility metadata becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does accessibility metadata hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=accessibility metadata; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain accessibility metadata? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Launch Plan - Operating Card 03

Core proposition

Launch plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by launch plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, launch plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does launch plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=launch plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain launch plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Analytics Loop - Operating Card 03

Core proposition

Analytics loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by analytics loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, analytics loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does analytics loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=analytics loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain analytics loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Platform Verification - Operating Card 03

Core proposition

Platform verification is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by platform verification.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, platform verification becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does platform verification hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=platform verification; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain platform verification? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Agent Roles - Operating Card 01

Core proposition

Agent roles is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent roles.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent roles becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent roles hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent roles; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent roles? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Tool Contract - Operating Card 01

Core proposition

Tool contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Draft Pipeline - Operating Card 01

Core proposition

Draft pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by draft pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, draft pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does draft pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=draft pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain draft pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Qa Pipeline - Operating Card 01

Core proposition

Qa pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QA pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QA pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QA pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QA pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QA pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Human Approval - Operating Card 01

Core proposition

Human approval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by human approval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, human approval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does human approval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=human approval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain human approval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Version Control - Operating Card 01

Core proposition

Version control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by version control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, version control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does version control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=version control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain version control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Ledger - Operating Card 01

Core proposition

Asset ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Prompt Library - Operating Card 01

Core proposition

Prompt library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by prompt library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, prompt library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does prompt library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=prompt library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain prompt library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Model Routing - Operating Card 01

Core proposition

Model routing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by model routing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, model routing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does model routing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=model routing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain model routing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Automation Guardrail - Operating Card 01

Core proposition

Automation guardrail is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by automation guardrail.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, automation guardrail becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does automation guardrail hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=automation guardrail; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain automation guardrail? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Dry-Run Rule - Operating Card 01

Core proposition

Dry-run rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dry-run rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dry-run rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dry-run rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dry-run rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dry-run rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Task Board - Operating Card 01

Core proposition

Task board is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by task board.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, task board becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does task board hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=task board; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain task board? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Handoff Format - Operating Card 01

Core proposition

Handoff format is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by handoff format.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, handoff format becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does handoff format hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=handoff format; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain handoff format? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Rollback Plan - Operating Card 01

Core proposition

Rollback plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rollback plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rollback plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rollback plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rollback plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rollback plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Incident Log - Operating Card 01

Core proposition

Incident log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by incident log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, incident log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does incident log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=incident log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain incident log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Agent Roles - Operating Card 02

Page 486 / 700 - P11 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Agent roles is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent roles.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent roles becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent roles hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent roles; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent roles? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Tool Contract - Operating Card 02

Core proposition

Tool contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Draft Pipeline - Operating Card 02

Core proposition

Draft pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by draft pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, draft pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does draft pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=draft pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain draft pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Qa Pipeline - Operating Card 02

Core proposition

Qa pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QA pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QA pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QA pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QA pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QA pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Human Approval - Operating Card 02

Core proposition

Human approval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by human approval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, human approval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does human approval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=human approval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain human approval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Version Control - Operating Card 02

Core proposition

Version control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by version control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, version control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does version control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=version control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain version control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Ledger - Operating Card 02

Core proposition

Asset ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Prompt Library - Operating Card 02

Core proposition

Prompt library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by prompt library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, prompt library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does prompt library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=prompt library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain prompt library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Model Routing - Operating Card 02

Core proposition

Model routing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by model routing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, model routing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does model routing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=model routing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain model routing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Automation Guardrail - Operating Card 02

Core proposition

Automation guardrail is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by automation guardrail.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, automation guardrail becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does automation guardrail hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=automation guardrail; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain automation guardrail? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Dry-Run Rule - Operating Card 02

Core proposition

Dry-run rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dry-run rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dry-run rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dry-run rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dry-run rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dry-run rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Task Board - Operating Card 02

Core proposition

Task board is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by task board.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, task board becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does task board hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=task board; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain task board? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Handoff Format - Operating Card 02

Core proposition

Handoff format is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by handoff format.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, handoff format becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does handoff format hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=handoff format; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain handoff format? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Rollback Plan - Operating Card 02

Core proposition

Rollback plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rollback plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rollback plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rollback plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rollback plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rollback plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Incident Log - Operating Card 02

Core proposition

Incident log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by incident log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, incident log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does incident log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=incident log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain incident log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Agent Roles - Operating Card 03

Page 501 / 700 - P11 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Agent roles is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent roles.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent roles becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent roles hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent roles; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent roles? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Tool Contract - Operating Card 03

Core proposition

Tool contract is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by tool contract.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, tool contract becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does tool contract hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=tool contract; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain tool contract? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Draft Pipeline - Operating Card 03

Core proposition

Draft pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by draft pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, draft pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does draft pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=draft pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain draft pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Qa Pipeline - Operating Card 03

Core proposition

Qa pipeline is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by QA pipeline.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, QA pipeline becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does QA pipeline hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=QA pipeline; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain QA pipeline? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Human Approval - Operating Card 03

Core proposition

Human approval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by human approval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, human approval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does human approval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=human approval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain human approval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Version Control - Operating Card 03

Core proposition

Version control is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by version control.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, version control becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does version control hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=version control; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain version control? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Asset Ledger - Operating Card 03

Core proposition

Asset ledger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by asset ledger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, asset ledger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does asset ledger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=asset ledger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain asset ledger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Prompt Library - Operating Card 03

Core proposition

Prompt library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by prompt library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, prompt library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does prompt library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=prompt library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain prompt library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Model Routing - Operating Card 03

Core proposition

Model routing is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by model routing.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, model routing becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does model routing hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=model routing; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain model routing? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Automation Guardrail - Operating Card 03

Page 510 / 700 - P11 - Evidence rank: Synthesis + operational protocol - Agent: Critique Agent

Core proposition

Automation guardrail is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by automation guardrail.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, automation guardrail becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does automation guardrail hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=automation guardrail; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain automation guardrail? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Dry-Run Rule - Operating Card 03

Core proposition

Dry-run rule is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by dry-run rule.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, dry-run rule becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does dry-run rule hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=dry-run rule; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain dry-run rule? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Task Board - Operating Card 03

Core proposition

Task board is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by task board.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, task board becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does task board hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=task board; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain task board? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Handoff Format - Operating Card 03

Core proposition

Handoff format is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by handoff format.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, handoff format becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does handoff format hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=handoff format; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain handoff format? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Rollback Plan - Operating Card 03

Core proposition

Rollback plan is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by rollback plan.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, rollback plan becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does rollback plan hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=rollback plan; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain rollback plan? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Incident Log - Operating Card 03

Core proposition

Incident log is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by incident log.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, incident log becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does incident log hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=incident log; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain incident log? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Chunk Schema - Operating Card 01

Core proposition

Chunk schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chunk schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chunk schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chunk schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chunk schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chunk schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Embedding Policy - Operating Card 01

Core proposition

Embedding policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by embedding policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, embedding policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does embedding policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=embedding policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain embedding policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Source Node - Operating Card 01

Page 518 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Source node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Node - Operating Card 01

Core proposition

Claim node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Concept Node - Operating Card 01

Page 520 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Concept node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Problem Node - Operating Card 01

Core proposition

Problem node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by problem node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, problem node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does problem node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=problem node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain problem node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Graph Relation - Operating Card 01

Page 522 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Graph relation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by graph relation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, graph relation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does graph relation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=graph relation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain graph relation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Community Summary - Operating Card 01

Page 523 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Community summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Local Query - Operating Card 01

Core proposition

Local query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by local query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, local query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does local query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=local query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain local query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Global Query - Operating Card 01

Page 525 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Global query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by global query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, global query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does global query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=global query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain global query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Retrieval Eval - Operating Card 01

Core proposition

Retrieval eval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by retrieval eval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, retrieval eval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does retrieval eval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=retrieval eval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain retrieval eval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Hallucination Guard - Operating Card 01

Core proposition

Hallucination guard is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination guard.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination guard becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination guard hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination guard; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination guard? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Update Trigger - Operating Card 01

Core proposition

Update trigger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by update trigger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, update trigger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does update trigger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=update trigger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain update trigger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Knowledge Graph - Operating Card 01

Core proposition

Knowledge graph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by knowledge graph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, knowledge graph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does knowledge graph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=knowledge graph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain knowledge graph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Agent Memory - Operating Card 01

Page 530 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Agent memory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent memory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent memory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent memory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent memory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent memory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Chunk Schema - Operating Card 02

Core proposition

Chunk schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chunk schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chunk schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chunk schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chunk schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chunk schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Embedding Policy - Operating Card 02

Page 532 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Embedding policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by embedding policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, embedding policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does embedding policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=embedding policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain embedding policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Source Node - Operating Card 02

Page 533 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Source node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Node - Operating Card 02

Core proposition

Claim node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Concept Node - Operating Card 02

Page 535 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Concept node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Problem Node - Operating Card 02

Core proposition

Problem node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by problem node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, problem node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does problem node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=problem node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain problem node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Graph Relation - Operating Card 02

Core proposition

Graph relation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by graph relation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, graph relation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does graph relation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=graph relation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain graph relation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Community Summary - Operating Card 02

Page 538 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Community summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Local Query - Operating Card 02

Page 539 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Local query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by local query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, local query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does local query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=local query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain local query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Global Query - Operating Card 02

Page 540 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Global query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by global query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, global query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does global query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=global query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain global query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Retrieval Eval - Operating Card 02

Page 541 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Retrieval eval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by retrieval eval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, retrieval eval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does retrieval eval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=retrieval eval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain retrieval eval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Hallucination Guard - Operating Card 02

Core proposition

Hallucination guard is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination guard.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination guard becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination guard hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination guard; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination guard? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Update Trigger - Operating Card 02

Core proposition

Update trigger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by update trigger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, update trigger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does update trigger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=update trigger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain update trigger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Knowledge Graph - Operating Card 02

Core proposition

Knowledge graph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by knowledge graph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, knowledge graph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does knowledge graph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=knowledge graph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain knowledge graph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Agent Memory - Operating Card 02

Page 545 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Agent memory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent memory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent memory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent memory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent memory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent memory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Chunk Schema - Operating Card 03

Core proposition

Chunk schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by chunk schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, chunk schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does chunk schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=chunk schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain chunk schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Embedding Policy - Operating Card 03

Core proposition

Embedding policy is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by embedding policy.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, embedding policy becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does embedding policy hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=embedding policy; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain embedding policy? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Source Node - Operating Card 03

Page 548 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Source node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by source node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, source node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does source node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=source node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain source node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Claim Node - Operating Card 03

Core proposition

Claim node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by claim node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, claim node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does claim node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=claim node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain claim node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Concept Node - Operating Card 03

Page 550 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Concept node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by concept node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, concept node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does concept node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=concept node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain concept node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Problem Node - Operating Card 03

Core proposition

Problem node is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by problem node.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, problem node becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does problem node hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=problem node; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain problem node? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Graph Relation - Operating Card 03

Core proposition

Graph relation is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by graph relation.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, graph relation becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does graph relation hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=graph relation; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain graph relation? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Community Summary - Operating Card 03

Page 553 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Community summary is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community summary.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community summary becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community summary hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community summary; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community summary? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Local Query - Operating Card 03

Page 554 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Local query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by local query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, local query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does local query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=local query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain local query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Global Query - Operating Card 03

Page 555 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Global query is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by global query.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, global query becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does global query hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=global query; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain global query? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Retrieval Eval - Operating Card 03

Page 556 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Tool Agent

Core proposition

Retrieval eval is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by retrieval eval.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, retrieval eval becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does retrieval eval hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=retrieval eval; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain retrieval eval? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Hallucination Guard - Operating Card 03

Core proposition

Hallucination guard is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination guard.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination guard becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination guard hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination guard; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination guard? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Update Trigger - Operating Card 03

Core proposition

Update trigger is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by update trigger.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, update trigger becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does update trigger hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=update trigger; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain update trigger? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Knowledge Graph - Operating Card 03

Core proposition

Knowledge graph is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by knowledge graph.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, knowledge graph becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does knowledge graph hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=knowledge graph; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain knowledge graph? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Agent Memory - Operating Card 03

Page 560 / 700 - P12 - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Agent memory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agent memory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agent memory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agent memory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agent memory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agent memory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Hallucination Court - Operating Card 01

Core proposition

Hallucination court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reductionism Court - Operating Card 01

Core proposition

Reductionism court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reductionism court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reductionism court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reductionism court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reductionism court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reductionism court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

False Novelty Court - Operating Card 01

Core proposition

False novelty court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by false novelty court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, false novelty court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does false novelty court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=false novelty court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain false novelty court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Weak Operationalization - Operating Card 01

Core proposition

Weak operationalization is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by weak operationalization.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, weak operationalization becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does weak operationalization hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=weak operationalization; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain weak operationalization? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Agentic Failure - Operating Card 01

Core proposition

Agentic failure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agentic failure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agentic failure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agentic failure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agentic failure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agentic failure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Reader Outcome - Operating Card 01

Core proposition

Reader outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Publishing Compliance - Operating Card 01

Core proposition

Publishing compliance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by publishing compliance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, publishing compliance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does publishing compliance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=publishing compliance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain publishing compliance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Medical Safety - Operating Card 01

Core proposition

Medical safety is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by medical safety.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, medical safety becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does medical safety hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=medical safety; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain medical safety? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Seo Spam Court - Operating Card 01

Core proposition

Seo spam court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO spam court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO spam court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO spam court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO spam court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO spam court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Copyright Court - Operating Card 01

Core proposition

Copyright court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Evidence Collapse - Operating Card 01

Core proposition

Evidence collapse is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence collapse.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence collapse becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence collapse hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence collapse; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence collapse? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Mcd Score - Operating Card 01

Core proposition

Mcd score is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD score.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD score becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD score hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD score; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD score? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Failure Code - Operating Card 01

Core proposition

Failure code is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by failure code.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, failure code becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does failure code hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=failure code; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain failure code? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Repair Protocol - Operating Card 01

Core proposition

Repair protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Survivability Index - Operating Card 01

Core proposition

Survivability index is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by survivability index.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, survivability index becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does survivability index hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=survivability index; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain survivability index? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Hallucination Court - Operating Card 02

Core proposition

Hallucination court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reductionism Court - Operating Card 02

Core proposition

Reductionism court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reductionism court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reductionism court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reductionism court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reductionism court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reductionism court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

False Novelty Court - Operating Card 02

Core proposition

False novelty court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by false novelty court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, false novelty court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does false novelty court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=false novelty court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain false novelty court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Weak Operationalization - Operating Card 02

Core proposition

Weak operationalization is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by weak operationalization.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, weak operationalization becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does weak operationalization hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=weak operationalization; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain weak operationalization? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Agentic Failure - Operating Card 02

Core proposition

Agentic failure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agentic failure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agentic failure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agentic failure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agentic failure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agentic failure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Reader Outcome - Operating Card 02

Core proposition

Reader outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Publishing Compliance - Operating Card 02

Core proposition

Publishing compliance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by publishing compliance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, publishing compliance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does publishing compliance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=publishing compliance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain publishing compliance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Medical Safety - Operating Card 02

Core proposition

Medical safety is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by medical safety.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, medical safety becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does medical safety hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=medical safety; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain medical safety? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Seo Spam Court - Operating Card 02

Core proposition

Seo spam court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO spam court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO spam court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO spam court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO spam court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO spam court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Copyright Court - Operating Card 02

Core proposition

Copyright court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Evidence Collapse - Operating Card 02

Core proposition

Evidence collapse is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence collapse.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence collapse becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence collapse hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence collapse; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence collapse? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Mcd Score - Operating Card 02

Core proposition

Mcd score is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD score.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD score becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD score hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD score; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD score? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Failure Code - Operating Card 02

Core proposition

Failure code is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by failure code.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, failure code becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does failure code hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=failure code; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain failure code? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Repair Protocol - Operating Card 02

Core proposition

Repair protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Survivability Index - Operating Card 02

Page 590 / 700 - P13 - Evidence rank: Synthesis + operational protocol - Agent: Skill Agent

Core proposition

Survivability index is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by survivability index.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, survivability index becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does survivability index hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=survivability index; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain survivability index? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Hallucination Court - Operating Card 03

Core proposition

Hallucination court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by hallucination court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, hallucination court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does hallucination court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=hallucination court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain hallucination court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Reductionism Court - Operating Card 03

Core proposition

Reductionism court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reductionism court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reductionism court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reductionism court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reductionism court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reductionism court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

False Novelty Court - Operating Card 03

Core proposition

False novelty court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by false novelty court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, false novelty court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does false novelty court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=false novelty court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain false novelty court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Weak Operationalization - Operating Card 03

Core proposition

Weak operationalization is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by weak operationalization.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, weak operationalization becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does weak operationalization hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=weak operationalization; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain weak operationalization? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Agentic Failure - Operating Card 03

Core proposition

Agentic failure is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by agentic failure.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, agentic failure becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does agentic failure hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=agentic failure; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain agentic failure? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Reader Outcome - Operating Card 03

Core proposition

Reader outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Publishing Compliance - Operating Card 03

Core proposition

Publishing compliance is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by publishing compliance.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, publishing compliance becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does publishing compliance hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=publishing compliance; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain publishing compliance? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Medical Safety - Operating Card 03

Core proposition

Medical safety is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by medical safety.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, medical safety becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does medical safety hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=medical safety; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain medical safety? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Seo Spam Court - Operating Card 03

Core proposition

Seo spam court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by SEO spam court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, SEO spam court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does SEO spam court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=SEO spam court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain SEO spam court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Copyright Court - Operating Card 03

Core proposition

Copyright court is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by copyright court.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, copyright court becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does copyright court hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=copyright court; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain copyright court? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Evidence Collapse - Operating Card 03

Core proposition

Evidence collapse is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evidence collapse.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evidence collapse becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evidence collapse hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evidence collapse; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S2.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evidence collapse? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S2 - W3C WCAG 2.2: contrast, structure, alternatives, navigation. Verification: Established accessibility standard; apply to PDF/EPUB/web sales pages.

Mcd Score - Operating Card 03

Core proposition

Mcd score is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by MCD score.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, MCD score becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does MCD score hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=MCD score; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S4.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain MCD score? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S4 - Google Search Central: helpful, reliable, people-first content and SEO basics. Verification: Official search guidance; not a guarantee of ranking.

Failure Code - Operating Card 03

Core proposition

Failure code is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by failure code.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, failure code becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does failure code hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=failure code; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S6.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain failure code? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S6 - Microsoft GraphRAG: knowledge-graph extraction, communities, summaries, global/local retrieval. Verification: Current architectural reference; implementation evolves.

Repair Protocol - Operating Card 03

Core proposition

Repair protocol is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by repair protocol.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, repair protocol becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does repair protocol hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=repair protocol; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S8.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain repair protocol? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S8 - Cognitive Load Theory literature: intrinsic/extraneous/germane load; working-memory limits. Verification: Strong instructional-design research family.

Survivability Index - Operating Card 03

Core proposition

Survivability index is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by survivability index.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, survivability index becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does survivability index hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=survivability index; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S10.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain survivability index? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S10 - Prior Moussaif HyperKB artifacts: MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers. Verification: User-owned theoretical synthesis; not external proof.

Benchmark Suite - Operating Card 01

Core proposition

Benchmark suite is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark suite.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark suite becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark suite hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark suite; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark suite? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Study - Operating Card 01

Core proposition

Reader study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

A/B Test - Operating Card 01

Core proposition

A/b test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by A/B test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, A/B test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does A/B test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=A/B test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain A/B test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Market Feedback - Operating Card 01

Core proposition

Market feedback is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market feedback.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market feedback becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market feedback hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market feedback; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market feedback? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Learning Outcome - Operating Card 01

Core proposition

Learning outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by learning outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, learning outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does learning outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=learning outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain learning outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Template Library - Operating Card 01

Core proposition

Template library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by template library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, template library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does template library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=template library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain template library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Product Line - Operating Card 01

Core proposition

Product line is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product line.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product line becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product line hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product line; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product line? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Niche Factory - Operating Card 01

Core proposition

Niche factory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by niche factory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, niche factory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does niche factory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=niche factory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain niche factory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Evergreen Update - Operating Card 01

Core proposition

Evergreen update is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evergreen update.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evergreen update becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evergreen update hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evergreen update; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evergreen update? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Translation Route - Operating Card 01

Core proposition

Translation route is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by translation route.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, translation route becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does translation route hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=translation route; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain translation route? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Course Conversion - Operating Card 01

Core proposition

Course conversion is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by course conversion.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, course conversion becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does course conversion hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=course conversion; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain course conversion? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Community Loop - Operating Card 01

Core proposition

Community loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Research Agenda - Operating Card 01

Core proposition

Research agenda is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research agenda.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research agenda becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research agenda hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research agenda; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research agenda? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Open Problems - Operating Card 01

Core proposition

Open problems is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by open problems.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, open problems becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does open problems hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=open problems; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain open problems? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Next Version - Operating Card 01

Page 620 / 700 - P14 - Evidence rank: Synthesis + operational protocol - Agent: Evidence Agent

Core proposition

Next version is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by next version.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, next version becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does next version hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=next version; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain next version? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Benchmark Suite - Operating Card 02

Core proposition

Benchmark suite is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark suite.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark suite becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark suite hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark suite; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark suite? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Study - Operating Card 02

Page 622 / 700 - P14 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Reader study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

A/B Test - Operating Card 02

Core proposition

A/b test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by A/B test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, A/B test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does A/B test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=A/B test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain A/B test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Market Feedback - Operating Card 02

Core proposition

Market feedback is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market feedback.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market feedback becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market feedback hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market feedback; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market feedback? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Learning Outcome - Operating Card 02

Core proposition

Learning outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by learning outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, learning outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does learning outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=learning outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain learning outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Template Library - Operating Card 02

Core proposition

Template library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by template library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, template library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does template library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=template library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain template library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Product Line - Operating Card 02

Core proposition

Product line is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product line.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product line becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product line hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product line; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product line? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Niche Factory - Operating Card 02

Core proposition

Niche factory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by niche factory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, niche factory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does niche factory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=niche factory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain niche factory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Evergreen Update - Operating Card 02

Core proposition

Evergreen update is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evergreen update.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evergreen update becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evergreen update hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evergreen update; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evergreen update? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Translation Route - Operating Card 02

Core proposition

Translation route is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by translation route.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, translation route becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does translation route hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=translation route; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain translation route? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Course Conversion - Operating Card 02

Core proposition

Course conversion is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by course conversion.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, course conversion becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does course conversion hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=course conversion; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain course conversion? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Community Loop - Operating Card 02

Core proposition

Community loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Research Agenda - Operating Card 02

Core proposition

Research agenda is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research agenda.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research agenda becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research agenda hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research agenda; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research agenda? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Open Problems - Operating Card 02

Core proposition

Open problems is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by open problems.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, open problems becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does open problems hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=open problems; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain open problems? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Next Version - Operating Card 02

Core proposition

Next version is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by next version.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, next version becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does next version hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=next version; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain next version? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Benchmark Suite - Operating Card 03

Core proposition

Benchmark suite is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by benchmark suite.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, benchmark suite becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does benchmark suite hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=benchmark suite; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain benchmark suite? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Reader Study - Operating Card 03

Page 637 / 700 - P14 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Reader study is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by reader study.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, reader study becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does reader study hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=reader study; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain reader study? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

A/B Test - Operating Card 03

Core proposition

A/b test is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by A/B test.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, A/B test becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does A/B test hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=A/B test; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain A/B test? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Market Feedback - Operating Card 03

Core proposition

Market feedback is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by market feedback.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, market feedback becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does market feedback hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=market feedback; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain market feedback? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Learning Outcome - Operating Card 03

Page 640 / 700 - P14 - Evidence rank: Synthesis + operational protocol - Agent: Destruction Court Agent

Core proposition

Learning outcome is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by learning outcome.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, learning outcome becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does learning outcome hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=learning outcome; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain learning outcome? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Template Library - Operating Card 03

Core proposition

Template library is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by template library.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, template library becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does template library hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=template library; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain template library? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Product Line - Operating Card 03

Core proposition

Product line is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by product line.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, product line becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does product line hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=product line; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain product line? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Niche Factory - Operating Card 03

Core proposition

Niche factory is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by niche factory.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, niche factory becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does niche factory hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=niche factory; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain niche factory? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Evergreen Update - Operating Card 03

Core proposition

Evergreen update is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by evergreen update.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, evergreen update becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does evergreen update hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=evergreen update; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain evergreen update? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Translation Route - Operating Card 03

Core proposition

Translation route is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by translation route.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, translation route becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does translation route hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=translation route; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain translation route? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Course Conversion - Operating Card 03

Core proposition

Course conversion is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by course conversion.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, course conversion becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does course conversion hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=course conversion; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain course conversion? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Community Loop - Operating Card 03

Core proposition

Community loop is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by community loop.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, community loop becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does community loop hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=community loop; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain community loop? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Research Agenda - Operating Card 03

Core proposition

Research agenda is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by research agenda.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, research agenda becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does research agenda hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=research agenda; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain research agenda? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Open Problems - Operating Card 03

Core proposition

Open problems is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by open problems.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, open problems becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does open problems hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=open problems; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain open problems? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Next Version - Operating Card 03

Page 650 / 700 - P14 - Evidence rank: Synthesis + operational protocol - Agent: Evidence Agent

Core proposition

Next version is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by next version.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, next version becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does next version hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=part; concept=next version; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain next version? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix A - Operational Glossary Cards

Core proposition

Operational glossary cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Operational Glossary Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Operational Glossary Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Operational Glossary Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Operational Glossary Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Operational Glossary Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix B - HyperSkills Cards

Core proposition

Hyperskills cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperSkills Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperSkills Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperSkills Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperSkills Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperSkills Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix C - HyperLoops Cards

Core proposition

Hyperloops cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperLoops Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperLoops Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperLoops Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperLoops Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperLoops Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix D - Agent Role Contracts

Core proposition

Agent role contracts is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Agent Role Contracts.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Agent Role Contracts becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Agent Role Contracts hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Agent Role Contracts; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Agent Role Contracts? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix E - RAG/GraphRAG Metadata Schema

Core proposition

Rag/graphrag metadata schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by RAG/GraphRAG Metadata Schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, RAG/GraphRAG Metadata Schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does RAG/GraphRAG Metadata Schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=RAG/GraphRAG Metadata Schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain RAG/GraphRAG Metadata Schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix F - Tool Algorithms

Core proposition

Tool algorithms is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Tool Algorithms.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Tool Algorithms becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Tool Algorithms hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Tool Algorithms; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Tool Algorithms? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix G - Publishing Checklists

Core proposition

Publishing checklists is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Publishing Checklists.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Publishing Checklists becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Publishing Checklists hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Publishing Checklists; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Publishing Checklists? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix H - Cover and Visual Briefs

Core proposition

Cover and visual briefs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Cover and Visual Briefs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Cover and Visual Briefs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Cover and Visual Briefs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Cover and Visual Briefs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Cover and Visual Briefs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix I - Book Rewriting Templates

Core proposition

Book rewriting templates is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Book Rewriting Templates.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Book Rewriting Templates becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Book Rewriting Templates hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Book Rewriting Templates; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Book Rewriting Templates? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix J - Final AGI OS Kernel

Page 660 / 700 - APP - Evidence rank: Synthesis + operational protocol - Agent: Research Agent

Core proposition

Final agi os kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Final AGI OS Kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Final AGI OS Kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Final AGI OS Kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Final AGI OS Kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Final AGI OS Kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix A - Operational Glossary Cards

Core proposition

Operational glossary cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Operational Glossary Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Operational Glossary Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Operational Glossary Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Operational Glossary Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Operational Glossary Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix B - HyperSkills Cards

Core proposition

Hyperskills cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperSkills Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperSkills Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperSkills Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperSkills Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperSkills Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix C - HyperLoops Cards

Core proposition

Hyperloops cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperLoops Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperLoops Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperLoops Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperLoops Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperLoops Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix D - Agent Role Contracts

Core proposition

Agent role contracts is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Agent Role Contracts.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Agent Role Contracts becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Agent Role Contracts hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Agent Role Contracts; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Agent Role Contracts? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix E - RAG/GraphRAG Metadata Schema

Core proposition

Rag/graphrag metadata schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by RAG/GraphRAG Metadata Schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, RAG/GraphRAG Metadata Schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does RAG/GraphRAG Metadata Schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=RAG/GraphRAG Metadata Schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain RAG/GraphRAG Metadata Schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix F - Tool Algorithms

Core proposition

Tool algorithms is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Tool Algorithms.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Tool Algorithms becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Tool Algorithms hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Tool Algorithms; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Tool Algorithms? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix G - Publishing Checklists

Core proposition

Publishing checklists is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Publishing Checklists.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Publishing Checklists becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Publishing Checklists hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Publishing Checklists; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Publishing Checklists? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix H - Cover and Visual Briefs

Core proposition

Cover and visual briefs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Cover and Visual Briefs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Cover and Visual Briefs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Cover and Visual Briefs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Cover and Visual Briefs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Cover and Visual Briefs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix I - Book Rewriting Templates

Core proposition

Book rewriting templates is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Book Rewriting Templates.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Book Rewriting Templates becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Book Rewriting Templates hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Book Rewriting Templates; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Book Rewriting Templates? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix J - Final AGI OS Kernel

Core proposition

Final agi os kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Final AGI OS Kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Final AGI OS Kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Final AGI OS Kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Final AGI OS Kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Final AGI OS Kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix A - Operational Glossary Cards

Core proposition

Operational glossary cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Operational Glossary Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Operational Glossary Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Operational Glossary Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Operational Glossary Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Operational Glossary Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix B - HyperSkills Cards

Core proposition

Hyperskills cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperSkills Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperSkills Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperSkills Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperSkills Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperSkills Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix C - HyperLoops Cards

Core proposition

Hyperloops cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperLoops Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperLoops Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperLoops Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperLoops Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperLoops Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix D - Agent Role Contracts

Core proposition

Agent role contracts is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Agent Role Contracts.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Agent Role Contracts becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Agent Role Contracts hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Agent Role Contracts; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Agent Role Contracts? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix E - RAG/GraphRAG Metadata Schema

Core proposition

Rag/graphrag metadata schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by RAG/GraphRAG Metadata Schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, RAG/GraphRAG Metadata Schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does RAG/GraphRAG Metadata Schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=RAG/GraphRAG Metadata Schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain RAG/GraphRAG Metadata Schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix F - Tool Algorithms

Core proposition

Tool algorithms is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Tool Algorithms.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Tool Algorithms becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Tool Algorithms hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Tool Algorithms; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Tool Algorithms? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix G - Publishing Checklists

Core proposition

Publishing checklists is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Publishing Checklists.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Publishing Checklists becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Publishing Checklists hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Publishing Checklists; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Publishing Checklists? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix H - Cover and Visual Briefs

Core proposition

Cover and visual briefs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Cover and Visual Briefs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Cover and Visual Briefs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Cover and Visual Briefs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Cover and Visual Briefs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Cover and Visual Briefs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix I - Book Rewriting Templates

Core proposition

Book rewriting templates is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Book Rewriting Templates.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Book Rewriting Templates becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Book Rewriting Templates hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Book Rewriting Templates; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Book Rewriting Templates? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix J - Final AGI OS Kernel

Page 680 / 700 - APP - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Final agi os kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Final AGI OS Kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Final AGI OS Kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Final AGI OS Kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Final AGI OS Kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Final AGI OS Kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix A - Operational Glossary Cards

Core proposition

Operational glossary cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Operational Glossary Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Operational Glossary Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Operational Glossary Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Agentic Versioning: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Operational Glossary Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Agentic Versioning; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Operational Glossary Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix B - HyperSkills Cards

Core proposition

Hyperskills cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperSkills Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperSkills Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperSkills Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. MCD Evaluation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperSkills Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=MCD Evaluation; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperSkills Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix C - HyperLoops Cards

Page 683 / 700 - APP - Evidence rank: Synthesis + operational protocol - Agent: Pedagogy Agent

Core proposition

Hyperloops cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperLoops Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperLoops Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperLoops Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Evidence Ranking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperLoops Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Evidence Ranking; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperLoops Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix D - Agent Role Contracts

Core proposition

Agent role contracts is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Agent Role Contracts.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Agent Role Contracts becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Agent Role Contracts hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Mechanism Diagramming: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Agent Role Contracts; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mechanism Diagramming; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Agent Role Contracts? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix E - RAG/GraphRAG Metadata Schema

Core proposition

Rag/graphrag metadata schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by RAG/GraphRAG Metadata Schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, RAG/GraphRAG Metadata Schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does RAG/GraphRAG Metadata Schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Rewrite Preservation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=RAG/GraphRAG Metadata Schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Rewrite Preservation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain RAG/GraphRAG Metadata Schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix F - Tool Algorithms

Core proposition

Tool algorithms is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Tool Algorithms.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Tool Algorithms becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Tool Algorithms hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. GraphRAG Chunking: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Tool Algorithms; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=GraphRAG Chunking; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Tool Algorithms? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix G - Publishing Checklists

Core proposition

Publishing checklists is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Publishing Checklists.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Publishing Checklists becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Publishing Checklists hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Publishing QA: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Publishing Checklists; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Publishing QA; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Publishing Checklists? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix H - Cover and Visual Briefs

Core proposition

Cover and visual briefs is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Cover and Visual Briefs.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Cover and Visual Briefs becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Cover and Visual Briefs hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Marketing Intent Mapping: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Cover and Visual Briefs; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Marketing Intent Mapping; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Cover and Visual Briefs? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix I - Book Rewriting Templates

Core proposition

Book rewriting templates is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Book Rewriting Templates.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Book Rewriting Templates becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Book Rewriting Templates hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Visual Hierarchy Control: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Book Rewriting Templates; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Visual Hierarchy Control; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Book Rewriting Templates? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix J - Final AGI OS Kernel

Core proposition

Final agi os kernel is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Final AGI OS Kernel.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Final AGI OS Kernel becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Final AGI OS Kernel hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Kernel Extraction: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Final AGI OS Kernel; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Kernel Extraction; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Final AGI OS Kernel? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Appendix A - Operational Glossary Cards

Core proposition

Operational glossary cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Operational Glossary Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Operational Glossary Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Operational Glossary Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Reader-Gradient Design: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Operational Glossary Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Reader-Gradient Design; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S3.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Operational Glossary Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S3 - Amazon KDP Help / Content Guidelines: formatting, manuscript, cover, AI disclosure. Verification: Current platform rule; must be rechecked at publishing time.

Appendix B - HyperSkills Cards

Core proposition

Hyperskills cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperSkills Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperSkills Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperSkills Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Kernel Extraction Loop: route the unit through generation, attack, repair, test, and memory. Page-Filling Layout: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperSkills Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Page-Filling Layout; loop=Kernel Extraction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S5.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperSkills Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S5 - Lewis et al. RAG 2020: parametric + non-parametric memory for knowledge-intensive generation. Verification: Research foundation for retrieval-grounded generation.

Appendix C - HyperLoops Cards

Core proposition

Hyperloops cards is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by HyperLoops Cards.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, HyperLoops Cards becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does HyperLoops Cards hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Problem Decomposition Loop: route the unit through generation, attack, repair, test, and memory. Source-to-Action Translation: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=HyperLoops Cards; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Source-to-Action Translation; loop=Problem Decomposition Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S7.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain HyperLoops Cards? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S7 - Nielsen Norman Group visual hierarchy: scale, contrast, grouping, hierarchy, scanning. Verification: UX research/practice reference for page hierarchy.

Appendix D - Agent Role Contracts

Core proposition

Agent role contracts is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by Agent Role Contracts.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, Agent Role Contracts becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does Agent Role Contracts hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Tool Construction Loop: route the unit through generation, attack, repair, test, and memory. Destruction Courting: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=Agent Role Contracts; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Destruction Courting; loop=Tool Construction Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S9.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain Agent Role Contracts? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S9 - OWASP Top 10 for LLM Applications 2025: prompt injection, output handling, agency, data poisoning. Verification: Security risk framework for agentic ebook pipelines.

Appendix E - RAG/GraphRAG Metadata Schema

Core proposition

Rag/graphrag metadata schema is accepted only when it changes what the writer, designer, publisher, or agent can do next. In this OS it is treated as a working unit with a boundary, mechanism, example, risk, evidence rank, and update path.

Deep mechanism

Input = raw idea, sources, reader level, market context, and platform constraint. Transformation = extract active difference, build mechanism, simplify language, place a visual block, attach a task, and send the unit to critique. Output = page-ready knowledge that can be read, retrieved, tested, and improved.

MCD pressure

Keep distinction density, integration across reader levels, transformation into action, agency, prediction, compression, and noise reduction. The concept fails when it is impressive but does not change behavior.

Operational protocol

1. Name the reader transformation produced by RAG/GraphRAG Metadata Schema.
2. Separate fact, interpretation, Moussaif synthesis, hypothesis, and speculative extension.
3. Compress the mechanism into one teachable law, then expand it into an example.
4. Design a full page: heading, body, table, callout, checklist, visual cue, and memory note.
5. Run Hallucination, Reductionism, False Novelty, and Weak Operationalization courts.
6. Write the RAG/GraphRAG metadata and decide the next update trigger.

Use-case transposition

For a technology ebook, RAG/GraphRAG Metadata Schema becomes an architecture diagram and lab. For a mental-health ebook, it becomes a safe psychoeducational worksheet with escalation boundaries. For a marketing ebook, it becomes a testable offer, hook, and conversion metric.

Seal: understanding + tool + exercise + test + memory.

Page asset

Asset: one title, one claim ledger row, one mechanism box, one worksheet prompt, one visual block, one mini-test, one failure code, one update memory.

Page production cell

Layer	Required output
source ledger	present / tested / repairable
mechanism	present / tested / repairable
example	present / tested / repairable
exercise	present / tested / repairable
visual hierarchy	present / tested / repairable
court	present / tested / repairable
metadata	present / tested / repairable
update trigger	present / tested / repairable

Destruction court + retrieval memory

Court: Attack: does RAG/GraphRAG Metadata Schema hide a claim without source? does it rename an old idea without operation? does it over-compress complexity? Repair: lower the confidence rank, add a boundary condition, attach an exercise, or delete the page until it produces operational effect.

Agentic Execution Loop: route the unit through generation, attack, repair, test, and memory. Mental-Health Safety Framing: train the capability with beginner exercise, advanced exercise, error list, and mastery signal.

RAG metadata: chunk_type=appendix; concept=RAG/GraphRAG Metadata Schema; confidence=theoretical-synthesis-or-supported; centrality=high; relation=operationalizes; skill=Mental-Health Safety Framing; loop=Agentic Execution Loop; risk=hallucination/reductionism/platform-drift; version=v2.0; source_anchor=S1.

Memory: Memory update fields: previous state, action taken, source rank, contradiction found, repair made, output produced, metric affected, next verification need.

Mini-test: Can a beginner explain RAG/GraphRAG Metadata Schema? Can a practitioner apply it today? Can an expert see the caveat? Can an agent retrieve the right chunk? Can a critic falsify the weak part?

Source anchor S1 - W3C EPUB 3.3 / EPUB Accessibility: publication format + accessibility metadata. Verification: Established standard; verify version before upload.

Source Ledger and Verification Queue

Core closure text

This page records the external and internal anchors that must be checked before publication. Platform rules, accessibility conformance, SEO guidance, RAG/GraphRAG practice, and LLM security risks can change. Prior Moussaif artifacts supply architecture and authorial synthesis, not independent empirical proof.

Execution checklist

1. Check KDP content, AI disclosure, manuscript, and cover rules before upload.
2. Validate PDF and EPUB exports; run accessibility and contrast checks.
3. Review Gumroad prohibited-products and refund policy before listing.
4. Record source version, retrieval date, contradiction, and next verification need.
5. Never present Moussaif synthesis as established external fact.

Verification / memory / deployment layer

Layer	Required output
S1 W3C EPUB 3.3 / EPUB Accessibility	publication format + accessibility metadata | Established standard; verify version before
S2 W3C WCAG 2.2	contrast, structure, alternatives, navigation | Established accessibility standard; apply to
S3 Amazon KDP Help / Content Guidelines	formatting, manuscript, cover, AI disclosure | Current platform rule; must be rechecked
S4 Google Search Central	helpful, reliable, people-first content and SEO basics | Official search guidance; not a
S5 Lewis et al. RAG 2020	parametric + non-parametric memory for knowledge-intensive generation | Research
S6 Microsoft GraphRAG	knowledge-graph extraction, communities, summaries, global/local retrieval | Current
S7 Nielsen Norman Group visual hierarchy	scale, contrast, grouping, hierarchy, scanning | UX research/practice reference for page
S8 Cognitive Load Theory literature	intrinsic/extraneous/germane load; working-memory limits | Strong instructional-design
S9 OWASP Top 10 for LLM Applications 2025	prompt injection, output handling, agency, data poisoning | Security risk framework for
S10 Prior Moussaif HyperKB artifacts	MCD/QMCD/MCAI/HAKIM-MCD, courts, loops, source ledgers | User-owned theoretical

Final memory rule: store outcomes, not vibes. Store failures with cause codes. Store source-version dependencies. Store why a decision was made, not only what was done. A loop that cannot be audited is not a HyperSkill; it is unmanaged automation.

Final AGI OS Kernel - Compact System Prompt

Core closure text

You are operating the Moussaif Ebook HyperKB OS. Transform any ebook topic into a serious, simple, research-grounded, applicable, pedagogical, visually coherent, agent-ready knowledge product. Declare material available, sourced, inferred, synthetic, speculative, and needing verification. Build domain constitution, extract generative kernel, build concept/problem/mechanism/skill/evidence graphs, write in 101 language while preserving mechanisms and evidence ranks, convert every claim into an action, tool, example, exercise, checklist, diagram, prompt, test, design block, RAG chunk, or publishing decision. Preserve valid substance when rewriting. Run destruction courts. Prepare deployment assets. Update memory after every failure.

Execution checklist

1. Paste before any ebook project.
2. Add topic, audience, promise, field, platform, risk tier, and source set.
3. Ask for source ledger, kernel, outline, page design, rewrite plan, and courts.
4. Require RAG/GraphRAG metadata and version memory.
5. End with measurable reader outcome and next experiment.

Verification / memory / deployment layer

Use this closure as a prompt, not as a substitute for evidence. Every new ebook must declare its material, source classes, assumptions, field risks, platform rules, reader outcome, and update rhythm. The OS succeeds only when it improves a manuscript and creates durable memory after failure.

Layer	Required output
Input	topic, audience, existing manuscript, sources, platform, risk tier
Process	kernel -> evidence -> mechanism -> 101 clarity -> design -> court -> deploy
Output	PDF/EPUB, workbook, cover brief, listing, RAG chunks, graph nodes, benchmark
Update	reader data, platform changes, source corrections, market feedback, failure logs

Final memory rule: store outcomes, not vibes. Store failures with cause codes. Store source-version dependencies. Store why a decision was made, not only what was done. A loop that cannot be audited is not a HyperSkill; it is unmanaged automation.

Final Operational Kernel - One Page Formula

Core closure text

Formula: Ebook OS = Reader Transformation + Source Ledger + Mechanism Graph + Pedagogical Compression + Visual Coherence + Exercises + Marketplace Deployment + RAG/GraphRAG Metadata + Destruction Courts + Version Memory. Final law: knowledge is complete only when it becomes operable; operability is complete only when it becomes skill; skill is complete only when it becomes loop; loop is complete only when it updates memory after contact with reality.

Execution checklist

1. Define the reader transformation.
2. Rank sources and expose uncertainty.
3. Extract the smallest kernel that can regenerate the domain.
4. Convert each claim into an asset and test.
5. Ship only after courts, QA, and update plan.

Verification / memory / deployment layer

Use this closure as a prompt, not as a substitute for evidence. Every new ebook must declare its material, source classes, assumptions, field risks, platform rules, reader outcome, and update rhythm. The OS succeeds only when it improves a manuscript and creates durable memory after failure.

Layer	Required output
Input	topic, audience, existing manuscript, sources, platform, risk tier
Process	kernel -> evidence -> mechanism -> 101 clarity -> design -> court -> deploy
Output	PDF/EPUB, workbook, cover brief, listing, RAG chunks, graph nodes, benchmark
Update	reader data, platform changes, source corrections, market feedback, failure logs

Final memory rule: store outcomes, not vibes. Store failures with cause codes. Store source-version dependencies. Store why a decision was made, not only what was done. A loop that cannot be audited is not a HyperSkill; it is unmanaged automation.

Construction Log and Honest Limits

Core closure text

This PDF is a dense 700-page HyperKB OS artifact generated from the request, prior Moussaif project structure, uploaded-file snippets, personal project memory, and current public sources. It is not a peer-reviewed empirical study, not medical or legal advice, and not a guarantee of KDP/Gumroad approval or sales. Real platform upload, EPUB validation, accessibility conformance, market performance, and reader learning outcomes require direct testing.

Execution checklist

1. Convert to Markdown chunks.
2. Create EPUB edition.
3. Create KDP/Gumroad listing assets.
4. Build RAG index and graph nodes.
5. Benchmark reader outcomes on a real manuscript.

Verification / memory / deployment layer

Use this closure as a prompt, not as a substitute for evidence. Every new ebook must declare its material, source classes, assumptions, field risks, platform rules, reader outcome, and update rhythm. The OS succeeds only when it improves a manuscript and creates durable memory after failure.

Layer	Required output
Input	topic, audience, existing manuscript, sources, platform, risk tier
Process	kernel -> evidence -> mechanism -> 101 clarity -> design -> court -> deploy
Output	PDF/EPUB, workbook, cover brief, listing, RAG chunks, graph nodes, benchmark
Update	reader data, platform changes, source corrections, market feedback, failure logs

Final memory rule: store outcomes, not vibes. Store failures with cause codes. Store source-version dependencies. Store why a decision was made, not only what was done. A loop that cannot be audited is not a HyperSkill; it is unmanaged automation.

Self-Reconstruction Memory / Next Iteration

Core closure text

Next iteration: feed a real manuscript into the rewriting engine, generate before/after revisions, test reader comprehension, measure action rate, collect objections, update the source ledger, and rebuild the weakest pages. The strongest ebook is not the longest; it is the one that teaches, tests, repairs, retrieves, and updates.

Execution checklist

1. Choose one real book or chapter.
2. Run preservation and rewrite court.
3. Test with readers or users.
4. Measure comprehension, action, trust, and conversion.
5. Update the OS and produce Version 3.

Verification / memory / deployment layer

Use this closure as a prompt, not as a substitute for evidence. Every new ebook must declare its material, source classes, assumptions, field risks, platform rules, reader outcome, and update rhythm. The OS succeeds only when it improves a manuscript and creates durable memory after failure.

Layer	Required output
Input	topic, audience, existing manuscript, sources, platform, risk tier
Process	kernel -> evidence -> mechanism -> 101 clarity -> design -> court -> deploy
Output	PDF/EPUB, workbook, cover brief, listing, RAG chunks, graph nodes, benchmark
Update	reader data, platform changes, source corrections, market feedback, failure logs

Final memory rule: store outcomes, not vibes. Store failures with cause codes. Store source-version dependencies. Store why a decision was made, not only what was done. A loop that cannot be audited is not a HyperSkill; it is unmanaged automation.