هندسة الهارنس الوكيلي نحو AGI
الجزء الثاني

أطلس عشرين Harness واقعية، حالة الفن 2026، وبنية ما بعد الهارنس لـ MOUSSAIFXT / AL RAZZI

دراسة كثيفة للأوراق والـrepositories والـbenchmarks والبروتوكولات والكتب التي تعيد تعريف الوكالة طويلة الأفق: من Prime Agent وClaude Code وCodex وHermes إلى Manus وDeep Agents وLetta وMicrosoft Agent Framework وNeMo Agent Toolkit.

RLM · Continual Harness · Context Engineering · Durable Execution · Agent Skills · MCP · A2A · ACP · AG-UI · Computer Use · World-Effect Verification · HyperKB · HyperSkill · HyperAbility · HyperLoop · MUT · MUG · MOUSSAIFXT

Zouhair Moussaif / زهير مساعف
Research & Engineering Synthesis
September 2026 · Arabic RTL · Amiri · A4 · 45 pages 05

منهجياً: «AGI-oriented» توصيف لاتجاه البحث والمعمارية، لا ادعاء بأن أي نظام في هذه الورقة - بما فيها MOUSSAIFXT - أثبت AGI. المصادر الخارجية تُفصل عن التركيبات الجديدة، وكل نظرية جديدة مرتبطة بآلية واختبار إبطال.

الجزء الثاني: من مقارنة الهارنس إلى علم هندسة الوكالة

PART II · REAL-WORLD HARNESS ATLAS · 2026 STATE OF THE ART

ينتقل هذا الجزء من سؤال «ما أفضل feature؟» إلى سؤال أشد صرامة: **ما البنية التي تجعل نظاماً وكيلياً قادراً على الاستمرار، العمل على بيئة حقيقية، استعادة حالته، التحقق من أثره، والتطور دون أن تتحول المرونة إلى انفلات؟** لذلك ندرس عشرين harness/framework عالية الإشارة، لا بوصفها قائمة شعبية ولا leaderboard للAGI، بل بوصفها مختبرات هندسية تكشف حلولاً مختلفة لمشكلات واحدة: السياق المحدود، التنفيذ طويل الأفق، العزل، الأدوات، الذاكرة، التفويض، الاستئناف، التقييم، والأمن.

أبرز تغير في 2026 هو أن harness لم يعد wrapper حول API. Prime Agent يضع persistent REPL + Continual Harness في المركز؛ Anthropic يفصل «الدماغ» عن «اليدين» ويعامل session log كذاكرة قابلة للاستجواب؛ Cursor وManus ينقلان الوكيل إلى VM كاملة؛ Deep Agents يضيف filesystem/context offloading/ subagents؛ Letta يجعل الذاكرة والهوية خبرة مستمرة؛ Microsoft وPydantic AI يعاملان durability/checkpointing تخاصية أصلية؛ NVIDIA يحول المراقبة والتقييم والتحسين إلى طبقة هندسية مستقلة. [W01-W20]

> **أطروحة الجزء الثاني**
>
> إذا كانت النماذج هي «قدرة احتمالية»، فإن الهارنس هو البنية التي تحولها إلى **وكالة قابلة للحساب والمراجعة.** وعليه فإن MOUSSAIFXT لا ينبغي أن يقلّد harness بعينه، بل أن يصبح sovereign microkernel يربط أفضل substrates تحت عقود موحدة للسلطة، السياق، الأدلة، الأثر، والاستعادة.

هذا البناء يواصل السلم الداخلي الموثق في HyperSkills → HKOS → MCD/QMCD/MCAI → Goal/Question/Difference → MOUSSAIF: Constitution → Verification/World Effect → Tools → Agents/Router → HyperLoops → HyperAbilities → Memory/Evolution، مع قاعدة أن الحالة المرصودة والتفويض والدليل والتحقق تتقدم على ثقة النموذج أو كثرة المخرجات. [M1-M3]

أربعة تحولات تستخلصها الأدبيات والrepos

الأول: state خارج الرسالة. Prime وLetta وMicrosoft وDeep Agents تتفق عملياً على أن الاستمرارية لا يمكن اختزالها في chat history؛ الحالة تصبح ملفات، DB، checkpoints، kernels أو logs يمكن استجوابها وإعادة تشغيلها. **الثاني: substrate جزء من الذكاء التشغيلي.** VM/container/browser/terminal ليست مجرد أدوات؛ شكلها يحدد ما يمكن إثباته وما يمكن rollback له.

الثالث: verification انفصل عن generation. من SWE-agent وClaude generator-evaluator إلى Courts الداخلية، يصبح المنتج والمراجع مسارين مختلفين. **الرابع: capability تتجه إلى packaging/protocols.** MCP/A2A/ACP/AG-UI/Agent Skills تجعل tool/agent/UX boundaries قابلة للتوصيل، وهو شرط لبناء MOUSSAIFXT كـmicrokernel لا كتطبيق مغلق.

```
Agency ≈ Model Prior × Harness State × Action Surface × Verification × Recovery × Environment Fidelity
```

هذه ليست معادلة علمية مثبتة، بل decomposition لاختبار السببية. قوتها أنها تجبرنا على سؤال: إذا ثبتنا خمسة عوامل وحذفنا السادس، هل يتغير world-effect؟ بهذا تنتقل الدراسة من «feature inventory» إلى علم تجريبي للهارنس.

المنهج: اختيار 20 نظاماً دون تحويل المقارنة إلى دعاية

Selection protocol · evidence discipline · architecture-first comparison

نستخدم أربعة شروط للاختيار: (1) وجود تنفيذ واقعي أو repository/docs يمكن فحصها؛ (2) وجود قرار معماري مميز يتجاوز prompt loop بسيط؛ (3) صلة مباشرة بمهام طويلة، coding/computer-use/research أو workflows متعددة الأدوات؛ (4) قابلية استخراج primitive يمكن اختباره داخل AL RAZZI/MOUSSAIFXT. تسمية «Top 20» هنا تعني **أعلى عشرين مرجعاً هندسياً في عينة هذا البحث**، لا ترتيباً مطلقاً للذكاء.

رتبة الدليل	ما نعتمد عليه	كيف يدخل الورقة
E-Primary	docs/repo/paper رسمي	وصف feature أو architecture كما يعلنها المشروع
E-Observed	benchmarks/evals مستقلة أو قابلة للتكرار	تقدير القدرة تحت بيئة محددة
E-Synthesis	اشتقاقنا المقارن	فرضية هندسية، لا حقيقة عن النظام الأصلي
E-Moussaif	مادة المشروع الداخلية	خريطة adoption أو نظرية جديدة قابلة للهدم

نمتنع عن دمج «model score» و«harness score» في رقم واحد. يمكن لهارنس قوي أن يبدو ضعيفاً مع نموذج أضعف، ويمكن لنموذج قوي أن يخفي عيوب harness. لذلك نقترح لاحقاً dual-axis evaluation: تثبيت النموذج ومقارنة harnesses، ثم تثبيت harness ومقارنة النماذج. ويضاف محور ثالث: البيئة نفسها، لأن Anthropic بيّنت أن اختلاف موارد البنية التحتية في agentic coding evals يمكن أن يغير النتائج عدة نقاط. [R15]

> **قاعدة causal adoption**
>
> لا نقتبس اسماً مثل «memory» أو «subagent». نبحث عن العلاقة السببية: ما المشكلة؟ ما primitive؟ ما تكلفة primitive؟ ما failure mode؟ ما الاختبار الذي يثبت أن تبنيه أفضل من baseline أبسط؟

عقد إعادة الإنتاج

كل مقارنة في HarnessBench-MX يجب أن تثبت: tool، container/VM image، repository snapshot، temperature/reasoning budget، model version، network policy، secrets policy، initial context، wall-clock budget، token/cost budget، retry policy، human intervention budget وgrader. بدون ذلك قد نقيس infrastructure noise لا architecture. هذا ينسجم مع أدبيات تقييم الوكلاء التي أظهرت حساسية النتائج للبيئة التنفيذية.

نضيف ثلاثة أنواع ablation: **component ablation** لحذف primitive، **authority ablation** لتغيير الصلاحيات دون تغيير النموذج، و**state ablation** لمسح الذاكرة/checkpoints. هذه الثلاثة تكشف هل التحسن صادر من معرفة محفوظة، قدرة تنفيذ، أم من النموذج نفسه.

```
Δ_harness = Score(model=m, env=e, H1) - Score(model=m, env=e, H0)
```

الخريطة الكبرى: 20 Harness/Framework عالية الإشارة

Master atlas · not an AGI leaderboard

#	النظام	الفكرة المميزة	ما نريد وراثته
1	Prime Agent	RLM + persistent IPython + Continual Harness	programmatic context + durable refinement
2	Claude Code / Agent SDK	hooks, subagents, sandbox, long-run harness	containment + evaluator separation
3	OpenAI Codex CLI	local agent + controlled execution	thin native CLI + secure workspace action
4	Hermes Agent	memory + skills + subagents + many runtimes	self-improving personal agent substrate
5	OpenCode	provider-agnostic + LSP + client/server	frontend/runtime decoupling
6	Gemini CLI	extensions + skills + subagents + policy	portable capability packaging
7	OpenHands	EventStream + sandbox runtime	event-sourced agent/runtime boundary
8	SWE-agent	Agent-Computer Interface	tool/interface design as capability multiplier
9	Aider	repo map + architect/editor split	structural code context
10	Cline	Plan/Act + checkpoints + teams	human-governed autonomy
11	Roo Code	modes + orchestrator + permission groups	role-specific authority surfaces
12	Continue	source-controlled AI checks	agent logic as versioned CI policy
13	Cursor Agents	isolated cloud VMs + long-running coding	full-dev-environment substrate
14	Manus	cloud computer + wide parallel agents	general VM + elastic agent society
15	Deep Agents	FS offload + summarization + subagents	context hierarchy
16	Letta Code	memory-first, git-backed MemFS, mods	portable identity + self-editing context
17	Microsoft Agent Framework	workflows + checkpoints + orchestrations	durable graph execution
18	CrewAI	Flows + persistent state + crews	explicit process layer
19	NVIDIA NeMo Agent Toolkit	profile/eval/optimize across frameworks	performance/evolution plane
20	smolagents	small code/tool loop + managed agents	minimal baseline discipline

الهدف من الجدول هو رؤية أن «الهارنس» عائلة من الاختيارات، من micro-loop إلى durable distributed runtime. [W01-W20]

تصنيف وظيفي للعشرين

General persistent agents
Hermes, Manus, Letta؛ identity/memory/substrate أوسع من coding.

Harness-native coding
Prime, Claude, Codex, OpenCode, Aider, SWE-agent, Cline, Roo, Cursor.

Meta-engineering
NeMo Agent Toolkit + protocols؛ تقيس وتربط وتحسن harnesses أخرى.

Workflow runtimes
OpenHands, Deep Agents, Microsoft AF, CrewAI؛ /event/state/graph durability في المركز.

نضيف **Zenith** كـcross-harness meta-orchestrator خارج العدّ الرسمي للعشرين: يشغّل Claude Code/Codex/Hermes عبر MCP/ACP، ويجعل repeated gap-finding، replanning، workers/testers والاستيقاف المنضبط محور long-horizon work. هذا مهم لـMUG لأن «الفرق النشط» يصبح control signal بين الجولات لا مجرد metric داخلي. [W22]

عشرة محاور للمقارنة: ما الذي يجعل Harness ناضجاً؟

Architecture profile instead of one-dimensional ranking

بدلاً من إعطاء «درجة AGI» اعتباطية، نستخدم profile بعشرة محاور. النضج ليس أكبر عدد features؛ بل قدرة المكونات على تشكيل causal loop يمكن قياسه. الأوزان المقترحة في تجاربنا المستقبلية: الاستمرارية 15، substrate والعزل 12، 12 context/knowledge، الأدوات والبروتوكولات 10، التفويض 10، التحقق 10، الأمن 10، الحياد تجاه النموذج 6، 9 observability/recovery، التعلم 6.

Execution Substrate

shell فقط أم container/VM/desktop/browser؟ هل filesystem/ network boundaries مستقلة؟ هل البيئة reproducible؟

State & Continuity

هل mission state خارج conversation؟ هل يوجد checkpoint/replay/ resume؟ هل يمكن قتل العملية ثم استئنافها دون إعادة التخمين؟

Delegation Topology

subagent أداة مؤقتة، manager-specialists، agent team، أم peer-to-peer remote A2A؟

Context & Memory

هل توجد hierarchy map/memory أم paging/offload/repo compaction فقط؟ وهل provenance محفوظ؟

Evolution

هل الفشل يتحول إلى memory/skill/prompt/config/training candidate؟ وهل promotion محكوم باختبار held-out؟

Verification

tests فقط، evaluator مستقل، functional world-state grader، أم evidence receipts ومحاكم؟

هذه المحاور تصبح schema في HarnessRegistry داخل MOUSSAIFXT، فيتم routing بالعقدة: مهمة codebase ضخمة قد تفضّل repo-map/context-native harness؛ مهمة UI طويلة قد تحتاج VM+browser؛ مهمة research شديدة الاتساع قد تستفيد من parallel society؛ مهمة عالية الأثر تحتاج أقوى containment/ verification حتى لو ارتفعت الكلفة.

من profile وصفي إلى tensor تجريبي

بدلاً من scalar ranking نقترح tensor: H[task_family, risk_class, horizon, substrate]. قد يتفوق harness في repo editing ويضعف في browser tasks؛ وقد تكون الذاكرة مفيدة بعد ساعات ومضرة في مهمة قصيرة. لذلك كل محور يقاس مشروطاً بطول المهمة وخطر الأثر.

```
Utility(H|τ)=wq·VerifiedSuccess - wc·Cost - wl·Latency - wr·Risk - wh·HumanInterventions + wt·TransferGain
```

الأوزان ليست ثابتة: في research نرفع evidence/coverage؛ في production deployment نرفع risk/recovery؛ في coding interactive نرفع latency. هذه «الحاكمية الوعائية» تترجم هنا إلى resource/metric routing لا استعارة.

1 · Prime Agent: الهارنس كبيئة برمجية مستمرة

Persistent REPL · Recursive Language Model · Continual Harness

Prime Agent من أوضح المشاريع التي ترفع harness من «حلقة tool calling» إلى بيئة حوسبة. وثائقه تصفه كـRLM-native coding and research harness مبني على persistent IPython kernel، recursive subagents، durable sessions، multi-process local runtime. في نموذج RLM يصبح prompt/context متغيرات يمكن للوكيل أن يفلترها ويبرمج فوقها بدلاً من حشرها كلها في رسالة. كما يحفظ Continual Harness ملاحظات subagent specs، skill descriptions، memories، prompt وتعديلات refinement كحالة متينة. [W01]

الأهم ليس Python نفسه؛ بل **تغيير شكل العلاقة بين النموذج والسياق.** النموذج لا «يقرأ الكون» دفعة واحدة، بل يشغّل كوداً لاختيار أجزاء منه، ينشئ subagents كنداءات function، ويحافظ على متغيرات خارج window. الورقة الرسمية المنشورة في أغسطس 2026 تجعل هذه الفكرة مركزية في long-horizon evaluation وتفصل بين strategy التي يبنيها النموذج وبين execution/recovery/resource accounting التي يفرضها harness. [R01]

> **Adopt into MOUSSAIFXT**
>
> نرث programmatic context وpersistent computational workspace، لكن canonical state يبقى خارج IPython kernel. أو JavaScript REPL يصبح driver يمكن استبداله، بينما MUG يملك authority وEvidence Ledger يملك closure.

اختبارنا: Corpus أكبر من نافذة النموذج بـ20x، ونقيس resident tokens، عدد restart recovery، subagent recursion، context slices، والأداء بعد إزالة persistent kernel. إذا لم ينجح ablation في إظهار فرق واضح، لا نرفع RLM إلى invariant مطلق.

تشريح معماري متقدم

الفرضية الهندسية: تحويل السياق إلى كائن برمجي داخل REPL مستمر، ثم فصل runtime عن طبقة Continual Harness التي تعدّل prompt notes والذاكرة والمهارات ومواصفات الوكلاء الفرعيين. [W01,R01]

موضع الحالة: المتغيرات والملفات وsession state لا تعتمد على إعادة سرد المحادثة؛ RLM يجعل الاسترجاع والتحويل والتفويض عمليات يمكن برمجتها. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: خطره أن تتراكم state قابلة للتعديل ذاتياً فتتحول إلى انحراف صامت أو skill غير صالح؛ لذلك refinement يجب ألا يساوي promotion. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Long-horizon repo task + kill/restart + context > window + held-out task completion, recovery loss, state bloat, false promotion؛ نقيس reuse.	داخل MOUSSAIFXT نأخذ الفصل بين RLM-runtime وCapability Genome، لكن نضيف Evidence Court وheld-out regression قبل ترقية أي تعديل.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Zenith + Reflexion + Voyager

Prime يربط التكيف بحالة harness قابلة للتحرير، بينما Zenith يركز على repeated gap-finding/stopping discipline، وReflexion/Voyager يقدمان ذاكرة feedback ومهارات قابلة لإعادة الاستعمال. التركيب المقترح هو أن refinement لا يكتب مباشرة إلى Stable state؛ بل ينتج Candidate يمر عبر Regression Court، وبذلك نفصل التعلم السريع عن السلطة طويلة الأمد.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

2 · Claude Code / Agent SDK: الاستقلالية داخل حدود

Sandbox · hooks · checkpoints · context engineering · evaluator separation

تجمع هندسة Claude Code عدة دروس يجب فصلها: sandbox بحدود filesystem/network، checkpoints قبل edits، hooks قبل/بعد الأدوات، subagents وbackground tasks، وملفات مشروع دائمة مثل CLAUDE.md/skills/rules. Anthropic تشير إلى أن sandbox يقلل permission prompts بصورة كبيرة لأنه ينقل الحماية من «موافقة لكل command» إلى boundary مسبق. هذه نقطة مهمة: **الأمان ليس مرادفاً لزيادة interruptions.** [W02]

في long-running work، تبنّى Anthropic أن compaction وحدها لا تكفي، وتستعمل handoff artifacts وgenerator/evaluator separation. وفي 2026 دفعت الفكرة أبعد نحو decoupling brain from hands: session log يحتفظ بسياق recoverable خارج window، بينما harness حر في تحويل slices قبل إعادة حقنها. هذا قريب جداً من MCAS/EVCF لدينا، لكنه يمنحنا قيداً جديداً: storage durable يجب ألا يُقيد طريقة context engineering المستقبلية. [R02-R04]

Moussaif extension

نحوّل hooks إلى MUG Policy Hooks، checkpoints إلى typed Recovery Journal موحد، evaluator إلى Court مستقل، وsession log إلى Event Ledger قابل للـpaging بواسطة MCD، لا summary أحادية.

التحفظ: كل scaffold قد يشيخ. Anthropic نفسها توثق أن context engineering المناسب يتغير مع تحسن النماذج؛ إذ يمكن حذف قدر كبير من system scaffolding عندما تصبح النماذج أكثر قدرة. لذلك سندخل Harness Staleness Ablation كاختبار دوري بدل تقديس prompt/hook قديم.

تشريح معماري متقدم

الفرضية الهندسية: الوكالة القوية تأتي من harness يوازن context engineering مع subagents، hooks، sandboxing وgenerator-evaluator loops، لا من prompt أكبر. [W02,R02-R05]

موضع الحالة: Anthropic تعامل session log ككائن durable يمكن للهارنس إعادة تشكيله، ما يسمح بفصل ذاكرة المهمة عن نافذة النموذج وبفصل brain عن hands. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: الـscaffolding الثابت يمكن أن يصبح قديماً مع تحسن النماذج؛ كما أن كثرة approvals قد تقتل throughput دون أن تعطي containment حقيقياً. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

مقارنة approval-heavy vs sandbox-heavy على نفس agent؛ measure interventions, unsafe attempts blocked, task success, latency, token overhead.

MOUSSAIF adoption

نأخذ hooks/sandbox/evaluator separation ونضعها تحت AuthorityGraph: كل tool له reversible boundary، network policy، capability lease، وpostcondition.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Context engineering + durable logs + containment

أدبيات Anthropic حول context engineering وlong-running harnesses تلتقي مع الأمن: إذا كان context مورداً محدوداً فلا ينبغي ملؤه بسجل التنفيذ الخام، وإذا كانت الأدوات خطرة فلا ينبغي حل المشكلة بطلب موافقة على كل خطوة. الحل هو session log خارجي قابل للاستعلام، summary/pointers متغيرة، sandbox بحدود filesystem/network، وevaluator منفصل يقرأ الأثر بعد التنفيذ.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

3 · OpenAI Codex CLI: النواة المحلية الرقيقة

Local coding agent · native CLI · configurable execution boundary

Codex CLI يقدم درساً مغايراً: لا تحتاج كل وكالة إلى runtime متخم. المستودع الرسمي يعرّفه كـcoding agent محلي يعمل في terminal، مع مسارات IDE/app/cloud فوق نفس الفكرة الأساسية. قيمة التصميم لمشروعنا هي **thin, native interaction surface**: executable قريب من filesystem وGit، قابل للتركيب في أدوات المطور بدلاً من فرض واجهة كبيرة. [W03]

من منظور MOUSSAIFXT، Codex يمثل driver ممتازاً لعقد Software Engineering، لا kernel حاكمي. نريد أن نستفيد من قربه من repository، طريقة packaging/configuration، وقابلية تشغيله من بيئات مختلفة، بينما نبقي HyperKB، Evidence Receipt، Mission Contract وCourts خارج driver. هذه الفصلية تمنع أن يصبح vendor-specific CLI هو «نظام التشغيل» نفسه.

Driver Contract

`invoke(task, workspace, permissions, context_handles) → event_stream + diff + commands + tests + artifacts`. لا تقبل MOUSSAIFXT كلمة «done» من driver. الإغلاق يحصل فقط بعد readback مستقل للحالة واختبارات الهدف.

نقترح مقارنة Codex/Claude/Prime/Hermes/OpenHands على نفس container/commit/model budget عبر Harbor. النتيجة التي نريدها ليست «من يفوز؟» فقط، بل attribution: هل الفرق من context management، sandbox، file interface، prompt scaffold، model، أم retry policy؟ هذا النوع من matched infrastructure هو شرط علمي قبل قبول وراثة أي ادعاء.

تشريح معماري متقدم

الفرضية الهندسية: نواة CLI محلية رقيقة تجعل repository وshell مساحة العمل الأساسية، وتفصل agent loop عن واجهة العرض. [W03]

موضع الحالة: القيمة ليست في terminal UI بذاته بل في جعل الأوامر، diffs، edits وtests آثاراً قابلة للفحص داخل workspace مألوف للمطور. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: إذا لم تُفصل authority عن convenience، يصبح الوصول الواسع إلى shell اختصاراً خطراً؛ وإذا تضخم client يصبح runtime صعب الاستبدال. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Same repo/same model across native driver and Codex driver؛ compare tool-call entropy, edit precision, rollback success, time-to-green-tests.

MOUSSAIF adoption

نحوّل هذا إلى DriverContract موحد: inspect→patch→execute→verify→receipt، بحيث يمكن استبدال Codex أو native AL RAZZI دون تغيير Mission DAG.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Codex + Terminal-Bench + SWE-bench

Codex يمثل harness coding محلياً؛ لكن تقييمه العلمي يتطلب environments مثل SWE-bench/Terminal-Bench كي لا نخلط جودة terminal UX بقدرة إصلاح حقيقية. MOUSSAIFXT لذلك يسجل كل driver تحت نفس benchmark protocol، ويمنع أي driver-specific privilege من تشويه المقارنة.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

4 · Hermes Agent: الذاكرة والمهارات كحلقة تعلم

Memory · Skills · Browser · Subagents · Schedules · Runtime portability

Hermes Agent مهم لمشروعنا لأنه يجمع في agent شخصي واحد ما نريد فصله إلى primitives قابلة للحكم: نفس core عبر CLI/TUI/messaging/desktop، ذاكرة بين الجلسات، skills تنشأ وتتحسن من الخبرة، بحث في المحادثات، terminal/browser، scheduled jobs، subagents، وprovider/runtime flexibility. وثائقه تؤكد progressive disclosure للskills حتى لا تُستهلك context بلا حاجة. [W04]

هذه البنية تتقاطع مباشرة مع HyperKB/HyperSkill/HyperLoop. الفرق الذي يجب الحفاظ عليه هو أن MOUSSAIF لا يعامل «memory learned by agent» كحقيقة معيارية تلقائياً. التعلم يجب أن يمر من provenance وconfidence وscope ثم verification. لذلك نقترح ثلاث طبقات: Experience Memory خام، Candidate Skill، ثم Promoted HyperSkill بعد replay/held-out eval.

ما نضيفه

rollback، authority caps، court promotion، Evidence lineage، cross-harness capability manifest.

ما نأخذه

skill creation، portable runtimes، scheduled autonomy، conversation search، same core across surfaces.

Hermes أيضاً يذكّر بأن desktop app ليست architecture منفصلة: UI يجب أن تكون client فوق headless runtime. هذه القاعدة مهمة لمشروع Windows MVP: AL RAZZI CLI وdesktop UI يستهلكان Command/Event API نفسه، فلا تصبح الواجهة مكاناً مخفياً للمنطق.

تشريح معماري متقدم

الفرضية الهندسية: حلقة تعلم عملية تجمع ذاكرة منتقاة، skills قابلة للإنشاء والتحسين، بحثاً في الجلسات، scheduling، subagents، وعدة terminal backends. [W04]

موضع الحالة: Hermes يفصل provider عن agent identity، ويوسع substrate من local إلى Docker/SSH/serverless، ما يجعله مرجعاً جيداً لقابلية النقل والاستمرارية. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: الذاكرة التي يكتبها النموذج قد تصبح authoritative بسرعة، والمهارات المتكاثرة قد تخلق تكراراً أو تعارضاً دون registry وقواعد inheritance. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Cross-session task family: هل skill جديد يقلل cost في المهمة التالية دون رفع regression؟ وهل backend migration يحفظ semantics نفسها؟

MOUSSAIF adoption

نحوّل memory/skills إلى HyperKB + HyperSkill with provenance, owner, scope, tests, expiry، ونستخدم Hermes كـexecution substrate لا كمصدر حقيقة أعلى.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Hermes + Agent Skills + Letta/Voyager

Hermes أقرب إلى terminal backends، scheduling، memory، skills: procedural lifelong assistant. هذا يتقاطع مع Voyager في executable skill library ومع Letta في memory-first identity. الفرق الذي نضيفه هو governance: كل memory/skill يجب أن تحمل source, scope, confidence, expiry، وtests كي لا تتحول الخبرة الذاتية إلى حقيقة سيادية.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

5 · OpenCode: فصل client عن server

Provider agnosticism · LSP · TUI · client/server architecture

OpenCode يقدم قراراً معمارياً بسيطاً لكنه عالي القيمة: client/server architecture مع TUI كأحد العملاء، provider agnosticism وLSP out of the box. النتيجة أن session/runtime لا يجب أن يرتبط بالواجهة التي تقوده. يمكن للعملية أن تعمل على جهاز بينما يقودها client آخر. [W05]

بالنسبة لـMOUSSAIFXTJ هذا يدعم مبدأ **Control Surface ≠ Cognitive Runtime**. نريد local daemon يحمل sessions/events/locks، بينما CLI/desktop/web/mobile مجرد control surfaces. LSP كذلك مثال على «tool ذو semantics عالية»: بدلاً من grep عام فقط، يستطيع harness الوصول إلى symbols, diagnostics, references بطريقة تقلل ضجيج السياق.

Extension: Semantic Tool Tier

نقسم tools إلى T0 text/raw، T1 structured filesystem/Git، T2 semantic code (LSP/AST), T3 world/browser/GUI، T4 high-impact external actions. MCD يفضّل أقل tier يعطي معلومة كافية؛ MUG يرفع متطلبات التفويض كلما ارتفع الأثر.

نرفض استنساخ UX بلا قياس. المعيار هو: هل فصل server/client يجعل reconnect/resume/multi-client أقل هشاشة؟ هل LSP يخفض tool calls/context tokens ويحسن repair precision؟ هذه تصبح تجارب لا شعارات.

تشريح معماري متقدم

الفرضية الهندسية: الفصل بين client وserver/provider يمنع تزاوج UX مع runtime ويتيح أكثر من surface فوق core واحد. [W05]

موضع الحالة: وجود LSP/provider-agnostic interfaces يجعل السياق البرمجي وأخطاء اللغة خدمات مستقلة عن نموذج بعينه. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: client/server split يضيف protocol/versioning burden؛ وإذا لم تكن contracts typed تصبح الأخطاء موزعة بين layers ويصعب attribution. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Replay نفس command trace عبر CLI وfuture UI، يجب أن تكون world effects والreceipts متطابقة باستثناء presentation events.

MOUSSAIF adoption

نأخذ headless-first architecture: Command API + Event API + typed state objects، ثم CLI/TUI/Desktop كلها projections على نفس kernel.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: ACP + client/server + provider neutrality

فصل client عن server يصبح أقوى حين يقترن بـACP: editor أو desktop لا يحتاج معرفة تفاصيل كل harness، بل يتعامل مع protocol. هذه البنية تدعم هدفنا UI-later: AL RAZZI CLI ليس واجهة نهائية بل أول client فوق control plane واحد.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

Gemini CLI: capability packaging + policy engine · 6

Extensions · Agent Skills · subagents · MCP · policy · sandbox

Gemini CLI أصبح مثالاً غنياً على packaging: extension قد يحمل prompts وMCP servers وcustom commands وhooks وsubagents وAgent Skills، مع Policy Engine وقواعد safety. Skills تكتشف من طبقات built-in/extension/user/workspace، وتفعّل on demand، بينما subagents تملك context مستقل وأدوات مخصصة؛ ويمكن للـsubagents remote استخدام A2A. [W06]

هذا يعطينا بنية واضحة لـCapability Genome: capability ليست prompt مفرداً بل package له version، permissions، assets، references، scripts، metadata، tests وHyperSkill dependencies. الحالي يجب أن يتوسع إلى manifest قابل للترجمة إلى Agent Skills حيث أمكن، مع الاحتفاظ بامتدادات MOUSSAIF الخاصة مثل evidence requirements وcourt gates.

> **نقطة ثورية عملية**
>
> نستخدم standard skill format كبوابة interoperability، لكن لا نجعل standard هو canonical ontology. HyperSkill → exporter → Agent Skill، والعكس يدخل كـcandidate غير موثوق حتى يجتاز validation/security scan.

كما أن consent عند activation في skills مثال جيد على capability-scoped authority: التفويض ليس «yes» global. يمكن منح skill وصولاً لمجلد وtoolset محددين لزمن محدد، ثم سحب السلطة عند نهاية node.

تشريح معماري متقدم

الفرضية الهندسية: Extensions تجمع prompts وMCP servers وcommands/hooks/subagents/skills، مع progressive disclosure وسياسات استخدام الأدوات. [W06]

موضع الحالة: subagent له سياق منفصل وأدوات متخصصة؛ ويمكن فصل capability packaging عن الحوار الرئيسي وتخفيف ضغط السياق. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: extension ecosystem قد يتحول إلى supply-chain attack surface أو إلى capability sprawl إذا لم يُوقّع ويُفحص ويُقيّد. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Malicious-extension injection + benign complex task؛ نقيس install-time detection, runtime containment, context savings, reproducibility.	نترجم extension إلى HyperPackage manifest: provenance, signature, required tools, network domains, secret scopes, tests, rollback recipe.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: MCP + Agent Skills + subagent isolation

Gemini يجمع ثلاثة اتجاهات معيارية: capability discovery عبر MCP، procedural packaging عبر Skills، وتفويض بسياق منفصل عبر subagents. هذا يبرر بناء HyperCapability manifests كطبقة مستقلة عن model/provider، لكن مع signatures وsecurity metadata إضافية.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

7 · OpenHands: EventStream كعمود فقري

Event-driven agent · Docker runtime · action/observation separation

OpenHands يعزل agent عن runtime عبر event stream. الـagent يولد Action، الـruntime ينفذه في sandbox ويرجع Observation، بينما backend/controller/ state ينسقان الدورة. Docker Runtime يهيئ shell/browser/plugins/Jupyter داخل بيئة معزولة وقابلة لإعادة الإنتاج. [W07]

هذا ينسجم مع AL RAZZI الحالي الذي يملك Command/Event contracts، لكنه يقترح تشديداً: **كل tool call يجب أن يصبح event durable قبل التنفيذ وبعده.** بهذه الطريقة يمكن independent verification، recovery، debugging، replay، وpost-hoc attribution دون الاعتماد على chat transcript.

```
IntentEvent → AuthorizationDecision → ToolInvocation
→ ObservationEvent → WorldReadback → EvidenceReceipt
→ CourtVerdict → StatePatch
```

الميزة المهمة هي أن runtime يمكن تغييره دون تغيير agent semantics. Docker اليوم، VM/WSL/remote sandbox غداً. MOUSSAIFXT يجب أن يعرّف interface ExecutionBackend لا «sandbox واحد». ويجب أن تحمل observations hashes/paths/timestamps بحيث لا تضيع العلاقة بين الفعل والأثر.

> **Ablation**
>
> قارن transcript-only logging مع event-sourced execution على crash mid-task: ما نسبة الاستئناف الصحيح؟ ما عدد الخطوات المعاد تنفيذها؟ وما القدرة على تحديد السبب بعد فشل صامت؟

تشريح معماري متقدم

الفرضية الهندسية: EventStream يجعل action/observation/events محور الفصل بين agent وruntime، بينما sandbox يحول التنفيذ إلى بيئة قابلة للرصد. [W07]

موضع الحالة: هذا النمط مهم لأن replay والتحليل لا يتطلبان إعادة بناء reasoning النصي؛ يكفي سجل أحداث typed مع state transitions. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ، ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: event log وحده ليس semantic evidence؛ قد يسجل أن command نجح بينما لم يتحقق الهدف، وقد يصبح الحجم كبيراً دون compaction/ pointers. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Fault injection at tool/runtime/event-store layers، measure replay fidelity، missing-event detection، causal attribution accuracy.	نجمع EventStream مع EvidenceReceipt: كل Action يرتبط preconditions، authority، Observation، postconditions، artifacts hashes، verifier verdict.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Event sourcing + durable workflow

EventStream يشبه event-sourced systems: الماضي ليس transcript بل سلسلة transitions. Microsoft checkpoints يكمل الفكرة من جهة snapshots. الجمع بين log + snapshot + world-effect receipt يعطي recovery أقوى من أي عنصر منفرد: replay للأسباب، snapshot للسرعة، receipt لمنع duplicate effects.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

8 · SWE-agent: واجهة الحاسوب جزء من الذكاء

Agent-Computer Interface · concise tools · linter-mediated edits

الدرس الأشهر في SWE-agent هو أن جودة Agent-Computer Interface تغيّر الأداء جذرياً. الورقة والوثائق تعرض أدوات مصممة للوكيل: search محدود، file viewer موجز، edit يمر على linter، ورسائل صريحة حتى عند command بلا output. هذه ليست تفاصيل UX؛ إنها **information shaping** يقلل الغموض ويمنع مسارات خطأ. [W08]

في MOUSSAIF نسمي هذا Tool Epistemics: كل tool يجب أن يحدد لا فقط schema للمدخل والمخرج، بل نوع المعرفة التي ينتجها، احتمال silent failure، درجة completeness، وverification strategy. مثال: grep يعثر على matches لكنه لا يثبت absence؛ test exit code يثبت شرطاً أضيق؛ UI screenshot دليل بصري لكنه قد لا يثبت backend state.

Tool Contract v2

purpose · preconditions · authority · input schema · observation semantics · failure modes · side effects · reversibility · verifier · cost · context footprint.

هذا يربط مباشرة بـMCD/QMCD: اختيار الأداة يصبح سؤالاً عن information gain وactionability وcost/noise، لا مجرد «هل لدى النموذج tool؟». نريد ToolGateway يفضل أدوات semantic عالية signal ويخفض tool spam، مع إمكانية query one level deeper عند uncertainty.

تشريح معماري متقدم

الفرضية الهندسية: Agent-Computer Interface يثبت أن شكل الأدوات والfeedback قد يرفع الأداء بقدر تغيير prompt/model؛ الواجهة جزء من القدرة. [W08]

موضع الحالة: أدوات بحث/تحرير concise، feedback من linter، وتحديد action space تقلل مساحة الخطأ مقارنة بـshell غير المنظم. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: ACI مصمم لمجال قد يفرط في تقييد agent عند الانتقال إلى UI/research/ops، أو يخفي قدرة نموذج لصالح tool semantics. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Ablate raw shell vs structured ACI على SWE-bench-like tasks؛ قِس invalid actions, recovery steps, patch correctness, context cost.

MOUSSAIF adoption

HyperAbility لا يساوي tool list: نعرّف capability surface لكل domain، مع minimal sufficient action vocabulary وescape hatch محكوم.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: ACI + interface design as intelligence

SWE-agent يذكرنا أن action vocabulary يمكن أن يكون inductive bias. عندما تكون الأدوات واضحة وfeedback قريباً من الفشل، يخفض search space. هذه الفكرة يجب أن تدخل HyperAbility Factory: توليد واجهة صغيرة متخصصة للمهمة قد يكون أفضل من إعطاء 80 tool عامة.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

9 · Aider: خريطة repository كضغط بنيوي

Repo Map · graph-aware code context · architect/editor split

Aider يستخدم repo map موجزة تشمل أهم classes/functions/signatures/critical lines عبر repository، لتمنح النموذج بنية الكود دون إدخال كل الملفات. هذه primitive تسبق موجة context engineering الحالية: compression لا يلخص النص فقط، بل يحافظ على **structure that predicts edit relevance**. كما يفصل Architect mode بين model يقترح الحل وeditor model يحوله إلى edits. [W09]

نوسع الفكرة إلى Project World Model: repo map ليست جدول symbols فقط، بل graph متعدد الطبقات: services, dependencies, tests, AST/LSP, filesystem, runtime ports, data stores, Git history، مع confidence وHyperKB freshness. لا يخزن duplicate كامل، بل pointers وsemantic summaries وتغيرات.

Moussaif extension

Architect = MUT/QMCD proposal؛ Editor = Execution driver؛ Verifier = Court/WorldEffect. لا يحق للمحرر إعلان صحة النظرية.

Architect ↔ Editor

فصل reasoning عن edit يقلل coupling؛ ويمكن جعل verifier ثالثاً لا يشارك generator context.

التجربة: نفس النموذج مع raw file listing مقابل repo map مقابل graph world model. نقيس dependency، wrong-file edits، token cost، task success، regressions، ووقت العثور على locus.

تشريح معماري متقدم

الفرضية الهندسية: repo map يضغط البنية الرمزية - classes/functions/signatures/relations - بدلاً من استدعاء ملفات عمياء، وarchitect/editor split يفصل تصميم التغيير عن صياغته. [W09]

موضع الحالة: الفكرة تعادل virtual memory للcodebase: لا تحتاج كل tokens الآن إذا امتلكت خريطة pointers تكشف أين توجد البنية ذات الصلة. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: الخريطة قد تصبح stale بعد edits أو تفشل في لغات/مولدات/monorepos؛ كما أن architect/editor قد يضيف handoff loss. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Scale codebase size 10× مع نفس context budget؛ measure symbol recall, wrong-file edits, map staleness, repair after refactor.

MOUSSAIF adoption

ندمج RepoMap في MCAS كـSymbol Plane مع invalidation بعد كل patch، ونربطه بـGitGraph وtests/evidence وليس summarization فقط.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Repo map + GraphRAG + virtual context

repo map هو GraphRAG مصغر للكود: العلاقات الرمزية هي أحياناً أهم من similarity النصية. في MCAS نستخدم AST/LSP/import graph/test graph كطبقة deterministic، ثم embeddings/semantic retrieval كطبقة لاحقة، كي لا نطلب من vector search اكتشاف بنية يمكن للمحلل اللغوي معرفتها بدقة.

ترجمة إلى **MUT/MUG**: MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

10 · Cline: الإنسان داخل الحلقة دون قتل الاستقلالية

Plan/Act · checkpoints · MCP · teams · schedules

Cline يجمع subagent spawning، checkpoints/undo، MCP، Plan/Act، agent teams، إضافة إلى scheduled/event-driven work في CLI. تصميمه مفيد لفهم human-in-the-loop كحالة تشغيلية لا كزر طوارئ فقط. [W10]

MOUSSAIFXT يحتاج Approval Semantics أغنى من allow/deny. نقترح: require-review، allow-until-state-change، allow-with-scope، allow-once-، deny-and-learn-policy، delegate-to-verifier، before-commit. كل قرار موافقة يصبح rule candidate يمكن أن يقلل friction لاحقاً لكن دون تحويل approval fatigue إلى auto-allow شامل.

Authority Lease

التفويض token مؤقت: {scope, tools, paths, domains, max_cost, expiry, reversibility, escalation}. subagent يرث الحد الأدنى فقط؛ لا يستطيع توسيع صلاحياته بنفسه.

هذه الفكرة تحقق MUG على مستوى التنفيذ. الاستقلالية ليست absence of human؛ هي قدرة النظام على التقدم داخل حدود معلومة، وعلى معرفة اللحظة التي يجب فيها السؤال لأن uncertainty أو irreversibility تجاوز threshold.

تشريح معماري متقدم

الفرضية الهندسية: MCP، checkpoints، Plan/Act، human approvals تقدم compromise عملياً بين autonomy وreviewability داخل IDE/CLI. [W10]

موضع الحالة: checkpointing يقلل تكلفة الرجوع ويجعل agent قابلاً للاستكشاف؛ plan mode يتيح مناقشة intent قبل side effects. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: إذا تحول Plan/Act إلى toggle شكلي، يبقى الهدف غير ممثل كـstate machine؛ والـapprovals قد تكون متكررة وغير مبنية على risk. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
User-interruption study: كم مرة يحتاج human intervention؟ وهل checkpoint rollback يعيد repo/state دون orphaned side effects؟	نستبدل Plan/Act binary بـAuthority Tiers T0-T4 + Effect Class، مع checkpoints آلية عند irreversible-risk boundaries.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Checkpoints + HITL + risk-based autonomy

Cline عملياً يوضح أن الإنسان يجب أن يظهر عند boundaries مهمة، لا كعامل تنفيذ دائم. لذلك نستبدل approval frequency بـrisk-triggered review: تغييرات reversible داخل branch تمر تلقائياً، secrets/deploy/delete/financial effects ترفع authority level أو تتطلب human gate.

ترجمة إلى MUT/MUG: MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

11 · Roo Code: modes كطبقات سلطة

Architect · Code · Debug · Orchestrator · custom permission groups

Roo Code يميز modes بوضوح، وOrchestrator لا يملك tools مباشرة بل يفوض عبر tasks. Custom modes تحدد tool groups، instructions، roleDefinition وfile restrictions. هذه الفكرة مناسبة لMoussaif لأننا نريد الفصل بين **من يقرر ومن يغير العالم.** [W11]

نقترح ترجمة modes إلى Authority Roles لا personas لغوية: Planner لا يكتب؛ Executor لا يغير goal؛ Verifier لا يصلح artifact الذي يحكم عليه؛ Security Court يستطيع veto؛ Researcher يستطيع اقتراح hypothesis لكنه لا يرفع claim rank. الأدوار تعرّفها capability/authority matrices وليس system prompt فقط.

Role	Write	Execute	Authorize	Verify
Planner	plan	لا	لا	جزئي
Executor	workspace	نعم	لا	self-check فقط
Verifier	evidence	read/test	لا	نعم
MUG Court	policy/state	لا	نعم/رفض	نعم

الفصل يحد من «self-certification»، وهو أحد أخطر أنماط الوكلاء: نفس السياق الذي ارتكب الخطأ يبرر نجاحه. Producer ≠ Verifier يصبح invariant.

تشريح معماري متقدم

الفرضية الهندسية: modes وOrchestrator يقدمان فكرة role-scoped permissions: architect/debug/code/orchestrate ليست personas فقط بل action surfaces مختلفة. [W11]

موضع الحالة: تفويض الوكيل الفرعي يصبح أكثر أماناً حين تحدد tool groups/file scopes بدلاً من prompt يقول كن حذراً. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: كثرة modes قد تتحول إلى taxonomy جامدة لا تكيف مع mission، وقد تتداخل الصلاحيات إذا لم تكن lattice واضحة. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier
measure unauthorized action؛ Privilege-escalation tests + delegation tasks attempts, permission deadlocks, completion under least authority.

MOUSSAIF adoption
mode names: role = {goal slice, read scope, write لا AuthorityGraph نستخدم scope, tools, budget, verifier, expiry}.

المعيار الحاكم: لا نرث branding أو UX، نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Modes → authority lattice

modes مفيدة لUX، لكن authority lattice هي الأعمق theory: صلاحيات القراءة والكتابة والتنفيذ والشبكة والسرية قابلة للترتيب والتقاطع. عند تفويض subagent نمنحه meet لأقل صلاحيات لازمة، لا persona واسعة.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

12 · Continue: الوكيل كسياسة versioned داخل CI

Source-controlled AI checks · repository-native governance

Continue أعاد توجيه المنتج نحو source-controlled AI checks: كل check ملف داخل repository ويمكن تشغيله على pull requests كحالة CI. الدرس ليس feature بعينه، بل **أن سلوك الوكيل يمكن أن يصبح جزءاً من source control** مثل tests وlint rules. [W12]

في MOUSSAIF نريد Governance-as-Code: constraints تُversion وتراجع عبر Git. هكذا يمكن تفسير لماذا تغير behavior بين إصدارين، والرجوع إلى policy سابق routing، Mission templates، HyperSkill manifests، Court rules، evaluator schemas، security policies

Policy Diff

أي تغير في prompt/skill/tool permissions/evaluator يخلق Behavioral Change Record مرتبطاً بـtests. لا يسمح بالترقية فقط لأن النص «أفضل»؛ يجب أن ينجح على regression corpus ويُظهر عدم تدهور safety.

هذا يحول self-evolution من تعديل غامض للذاكرة إلى promote → canary → court → eval → candidate branch: software lifecycle. وهو جسر مباشر بين HyperSkill Factory وDevOps.

تشريح معماري متقدم

الفرضية الهندسية: نقل AI checks وagent configuration إلى source control يجعل السلوك جزءاً من software lifecycle لا إعداداً محلياً غير مرئي. [W12]

موضع الحالة: versioning يسمح بمراجعة تغييرات القواعد كما تراجع code، وبربط agent policy بـCL وbranch/release. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: static checks قد تتحول إلى prompt linting غير سببي، وقد تصبح policy bundle كبيرة فتكرر ما يفعله tests/security tools أصلاً. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier: Policy mutation testing: غيّر rule عمداً واختبر أن CI/evals تكشف regression قبل promotion إلى runtime.

MOUSSAIF adoption: نأخذ policy-as-code: Courts/HyperLoops/quality gates تُخزن كـversioned manifests مع owners, test fixtures, deprecation and migration.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Policy-as-code + evaluation-as-code

وضع checks/config في Git يسمح بأن تصبح governance قابلة للمراجعة والrollback. داخل MOUSSAIF نوسع ذلك إلى Court-as-code وBenchmark-as-code: التغيير في rule أو verifier نفسه يخضع pull request واختباراً مثل أي code.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

13 · Cursor: الوكيل داخل بيئة تطوير كاملة

Cloud Agents · isolated VMs · multi-repo · browser/desktop

Cursor Cloud Agents تعمل في VMs مع startup commands، secrets، dependencies، cloned repos، وnetwork، ويمكنها build/test/interact مع software والـdesktop/browser. كما تدعم multi-repo environments والعمل المتوازي. وفي long-running preview تصف Cursor harness مخصصاً لمعالجة فشل النماذج في المهام طويلة الأفق. [W13]

هذا يدعم نظرية Substrate Completeness: بعض المهام لا تحل بأدوات RPC ضيقة، بل تحتاج بيئة Turing-complete قريبة من بيئة الإنسان. لكن VM كاملة تزيد blast radius والكلفة؛ لذلك يجب ألا تكون default. MCD router يقرر متى تبرر المهمة الانتقال من shell sandbox إلى full VM.

Tiered Substrate

T0 no-exec → T1 read-only FS → T2 sandbox shell → T3 container with services → T4 full VM/browser/desktop → T5 external-world actions. كل انتقال يحتاج evidence أن tier الأدنى غير كافٍ.

كما نرث فكرة artifacts of execution: screenshots, logs, videos, diffs ليست مجرد debugging؛ يمكن أن تدخل Evidence Ledger كأدلة متعددة الوسائط، ثم تتحقق Courts من علاقتها بالهدف.

تشريح معماري متقدم

الفرضية الهندسية: Cloud Agents تنقل الوكيل من chat مرتبط بـIDE إلى machine كاملة مع repo/dependencies/secrets/start commands/network، وتسمح بالعمل المتوازي. [W13]

موضع الحالة: الـVM هنا ليست مجرد sandbox: إنها memory substrate للعمل طويل الأفق ويمكنها تشغيل build/test/browser/desktop بشكل متزامن. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: VM persistence يوسع attack surface وcost؛ parallel agents قد يضاعفون work أو ينتجوا conflicting branches دون integration court. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Parallel feature task across N VMs؛ measure speedup, merge conflicts, idle cost, secret exposure surface, reproducibility from clean image.

MOUSSAIF adoption

يجب أن يعتبر Moussaif Harness Driver VM lease مورداً محكوماً: image hash, network policy, secrets lease, budget, branch, cleanup receipt.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Cloud VM + parallel branches

Cursor يبرز أن long-horizon coding يحتاج environment كاملة لا مجرد remote shell. لكن parallel VM work يجب أن يدار كبنية worker branch/merge: لا يكتب إلى Integration Court، main يقيس conflicts/tests/evidence قبل الدمج، وVM image hash يدخل في reproducibility record.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

14 · Manus: الحاسوب السحابي والمجتمع الوكيلي المرن

Persistent cloud computer · Wide Research · general-purpose subagents

Manus مهم لأنه يتعامل مع الجلسة كحوسبة سحابية كاملة. Cloud Computer في 2026 يضيف VM دائمة بـfilesystem وprocesses تعمل بين الجلسات، بينما Wide Research يحول المهمة إلى parallel processing مع agent-to-agent collaboration؛ subagents ليست أدواراً ثابتة بل نسخاً general-purpose. [W14]

هذه القوة تكشف أيضاً خطراً: parallelism ليس intelligence تلقائياً. إذا قسم الوكيل المهمة خطأً، نحصل على 20 مساراً مكلفاً من الخطأ. لذلك تقترح MOUSSAIF Delegation Value Test: لا spawn إلا إذا كان expected information gain أو wall-clock benefit أعلى من coordination/context/evaluation cost.

Bounded Agent Society

كل agent clone له evidence duty، budget، authority lease، context namespace، mission slice، communication، termination condition. بين peers events موثقة، ولا يجوز أن تتحول majority vote إلى proof.

نستخدم Wide Research كمرجع لـelasticity، ثم نضيف QMCD لتحديد branches ذات أعلى قيمة تمييزية، وCourts لتجنب «consensus hallucination» حين تعيد agents نفس الافتراض من مصدر واحد.

تشريح معماري متقدم

الفرضية الهندسية: Cloud Computer دائم + Wide Research يجعل general-purpose VM ومجتمع وكلاء مرن substrate للبحث والتنفيذ لا مجرد remote browser. [W14]

موضع الحالة: subagents عامة الغرض يمكن تخصيصها ديناميكياً بدلاً من قائمة ثابتة؛ هذا يقرّب society من task-adaptive compute allocation. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: التوازي الواسع يهدد الاتساق: runaway cost، conflicting evidence، duplication، وصعوبة إسناد النتيجة إلى العامل المؤثر. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Breadth-scaling curve: 1/4/16/64 workers على research task؛ قِس marginal information gain, redundancy, synthesis errors, cost per verified fact.

MOUSSAIF adoption

نأخذ elastic society لكن تحت MCD/QMCD admission: لا spawn إلا إذا expected information gain أو parallelism benefit يتجاوز overhead.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

Society كـauthority topology

عدد agents ليس metric نضج. topology الأنسب قد تكون peer swarm، manager→specialists، producer→verifier، أو blackboard. القرار يتعلق بكمية المعلومات القابلة للتوازي، coupling بين subtasks، تكلفة الدمج وخطر تضارب السلطة. في meta-governor، MUG لا يكتب كل شيء؛ يحدد من يملك أي claim/action ومدة التفويض.

```
Spawn if E[ParallelGain + InformationGain + VerificationGain] > CoordinationCost + MergeRisk + ContextCost
```

كل delegation يحمل lease منتهي الصلاحية وعودة «done» «structured: findings, evidence refs, artifacts, confidence, unresolved gaps. من subagent لا يغلق المهمة؛ Closure Court يراجع global invariants.

صلة بحثية: Wide Research + dynamic society

Wide Research يحول التوازي من fixed roles إلى تخصيص agents عامة بحسب subtasks. هذا قريب من compute routing: نريد elastic pool، لكن MUG يبقى meta-governor يمنع انفجار العدد. كل worker يعود بحزمة evidence قابلة للدمج لا prose طويل فقط.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

15 · Deep Agents: السياق كهرمية تخزين

Filesystem offload · summarization · context isolation · pluggable backends

Deep Agents يصرح بأنه agent harness للمهام متعددة الخطوات، مع long-term memory، subagents، filesystem، planning، وdurable LangGraph runtime. أكثر قرار يهمنا هو context management: large tool inputs/results تُoffload إلى filesystem وتستبدل بمراجع، الرسائل القديمة تُلخص؛ subagents تعزل العمل الثقيل؛ backends قابلة للتبديل بين memory/local/store/sandbox/composite. [W15]

هذه عملياً نسخة هندسية من فرضية EVCF/MCAS لدينا: model window ليست الذاكرة الوحيدة. لكننا نوسعها إلى **multi-tier cognitive memory**: L0 prompt/working set، L1 pinned summaries، L2 event/session store، L3 HyperKB/graph/symbols، L4 artifacts/files، L5 cloud archives. ContextPager يختار page بناء على Active Difference وprovenance/freshness.

> **Context Fault**
>
> إذا واجه الوكيل معلومة غير resident لا يخمن. يصدر ContextFault(resource_class, query, reason, urgency)، فيبحث pager ويعيد capsule مع source handles. failure to retrieve يبقى uncertainty صريحة.

الاختبار الأساسي ليس «كم GB لدينا؟»، بل useful information per resident token وكم مرة أدى paging إلى تصحيح قرار كان سيصبح خاطئاً.

تشريح معماري متقدم

الفرضية الهندسية: context engineering صريح: durable execution، long-term memory، subagent isolation، summarization، filesystem offload، وHITL فوق LangGraph. [W15]

موضع الحالة: المحتوى الكبير ينقل خارج نافذة النموذج إلى files/pointers؛ subagents تملك contexts مستقلة تمنع تلوث السياق الرئيسي. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: summaries قد تمحو تفاصيل causally important؛ filesystem offload دون indexing/paging policy قد يصبح مجرد dump كبير. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Context-stress suite أكبر 20× من window؛ measure retrieval miss, summary-loss, tool re-reads, decision accuracy under fixed token budget.	MCAS يضيف tiered pages: hot working set, symbol map, episodic trace, HyperKB facts, cold artifacts؛ كل page له provenance وeviction reason.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Context engineering as storage hierarchy

Deep Agents يجعل filesystem أداة context. التطوير الذي نقترحه هو أن التخزين نفسه يصبح typed: fact page، symbol page، artifact page، episode page، decision page. هذا يتيح eviction/retrieval policies مختلفة ويمنع أن تصبح الملفات سلة واحدة غير قابلة للحكم.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

16 · Letta Code: الذاكرة كهوية قابلة للنقل

Git-backed MemFS · self-editing memory · skills · mods · dreaming

Letta Code يضع memory في صلب agent identity: MemFS git-backed، memory blocks داخل السياق، external memory وskills خارج السياق، subagents للـrecall/reflection، وmods لتغيير harness. يميز صراحة بين ما يجب أن يتعلمه agent في memory، ما يجب أن يصبح skill إجرائية، وما يجب أن يفرضه harness كـpermission/hook/tool/config. [W16]

هذا التمييز شديد القيمة لـMOUSSAIF: **المعرفة ليست كلها من نوع واحد.** نقترح Memory Ontology: semantic facts، episodic traces، procedural skills، identity/preferences، policy/governance، world state، hypotheses. كل نوع له retention/verification/access مختلف.

Memory Promotion Ladder

trace خام → extracted episode أو HyperSkill → generalized candidate → contradiction search → replay → provenance binding → HyperKB fact. Policy proposal لا يحق لـreflection agent أن يكتب مباشرة في constitutional memory.

Git-backed memory كذلك يسمح diff/revert/audit، ما ينسجم مع Self-Reconstruction Memory. لكن commit history ليس truth؛ هو provenance. يجب أن تظل الأدلة الخارجية منفصلة عن «ما قرر الوكيل أن يتذكره».

تشريح معماري متقدم

الفرضية الهندسية: memory-first agent يجعل الهوية والذاكرة state مستمرة قابلة للنقل، مع سياق يمكن للوكيل تعديله بدلاً من أن يكون prompt ثابتاً. [W16]

موضع الحالة: التمييز المهم هو أن memory ليست retrieval لاحق فقط؛ إنها جزء من control state الذي يعيد تشكيل سلوك الجلسات القادمة. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: self-editing memory قد تؤدي إلى preference drift أو accidental forgetting؛ كما أن الذاكرة العامة قد تلوث مشاريع مختلفة. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Cross-project contamination test + memory edit rollback؛ قِس transfer benefit مقابل negative transfer وability to reconstruct why a memory exists.

MOUSSAIF adoption

HyperMemory تنقسم إلى user / learned / episodic / project / constitutional-skill، وكل طبقة لها write authority وTTL/provenance مختلف.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Memory identity + reconstruction

الذاكرة الدائمة لا تكفي؛ يجب أن نعرف لماذا توجد. لذلك Self-Reconstruction Memory يحفظ causal link من memory إلى evidence/outcome. عند تغير المشروع يمكن invalidation عبر dependency graph بدلاً من الاعتماد على model ليقرر أن معلومة قديمة.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

17 · Microsoft Agent Framework: durability كحالة نظام

Graph workflows · checkpoints · HITL · sequential/concurrent/handoff/magentic

Microsoft Agent Framework (1.0) في 2026 يوحّد production agents/workflows مع أنماط sequential, concurrent, handoff, group chat وMagentic. الأهم هو checkpointing: snapshot لحالة executors والرسائل pending والrequests/shared state في نهاية supersteps، ما يسمح pause/resume والمهاجرة بين instances. [W17]

هذا يدعم نظرية Interruption-Resilient Agency: الاستمرارية ليست replay chat بعد crash، بل state machine لها durable checkpoints. AL RAZZI يجب أن يستطيع kill -9 في منتصف mission ثم resume من آخر transition idempotent، مع تمييز actions التي حدث أثرها في العالم عن actions التي لم يحدث أثرها.

```
checkpoint = {mission, dag, node_states, pending_events,
leases, context_handles, world_receipts, budgets, version}
```

مشكلة exactly-once صعبة خصوصاً للأفعال الخارجية. الحل ليس الادعاء بها دائماً، بل idempotency keys + effect receipts + readback + compensating action. هذا يربط workflow durability بـWorldEffect، لا بـstate التخطيط فقط.

تشريح معماري متقدم

الفرضية الهندسية: workflow checkpoints تحفظ pending messages/requests/shared state، executor state، ما يجعل pause/resume/migration جزءاً من نموذج التشغيل. [W17]

موضع الحالة: checkpoint عند superstep يعيد تعريف durability من transcript إلى snapshot كامل للحالة التنفيذية. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: snapshot المتين قد يعيد تشغيل side effect مرتين إذا لم تكن الأدوات idempotent أو إذا غاب effect ledger. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Crash matrix at every superstep boundary، measure duplicate effects, state divergence, recovery time, replay determinism.	نضيف Exactly-Once-ish Effect Protocol: idempotency key + precondition hash + receipt + compensation handler قبل checkpoint commit.

المعيار الحاكم: لا نرث branding أو UX، نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Checkpoints + idempotency + migration

checkpoint الكامل يسمح migration بين instances، لكنه يفرض سؤال distributed transactions: هل effect الخارجي دخل checkpoint أم لا؟ Evidence-Carrying Action يحل فجوة exactly-once عملياً باستخدام idempotency keys، receipts والاستعلام عن world state قبل retry.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

18 · CrewAI: العملية الصريحة فوق المجتمع الوكيلي

Flows · state persistence · crews · guardrails · observability

CrewAI يفصل Agents/Crews عن Flows التي تدير persistence، state، start/listen/router، واستئناف workflows، مع /guardrails/knowledge/memory observability. هذا مثال على ضرورة وجود **process layer صريحة** بدلاً من ترك كل orchestration في chain-of-thought مبطن. [W18]

في MOUSSAIF، HyperLoop يجب أن يصبح graph contract لا وصفاً نصياً: stop conditions، retries، timeouts، branch conditions، invariants، inputs، compensation وHyperAgent. evidence gates ينفذ عقدة، لكنه لا يمتلك topology بالكامل إذا لم يحتجها.

> **Flow Minimality**
>
> لا نستخدم multi-agent إذا كان workflow deterministic كافياً. الوكالة تُحجز للمواضع التي تحتاج judgement/uncertainty. هذه القاعدة تخفض التكلفة وتزيد قابلية التحقق.

بهذا يمكن للنظام أن يخلط deterministic executors وagents في DAG نفسها: parse/compile/test خطوات ثابتة، بينما diagnosis/design/research nodes وكيلية. هذا أقرب إلى نظام تشغيل من «مجلس agents يتحدث».

تشريح معماري متقدم

الفرضية الهندسية: التفريق بين Crews وFlows: مجتمع الوكلاء ينفذ الأدوار، بينما Flow يمسك state/process/event routing ويمكنه الاستئناف. [W18]

موضع الحالة: هذه طبقة process explicit تمنع أن تكون orchestration سلسلة prompts مبهمة؛ state يمكن ملاحظته وربطه بمسار. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: role-play semantics قد تكون أضعف من typed contracts؛ كثرة agents قد تخفي workflow خطياً كان يمكن تنفيذه بأداة واحدة. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier
Single-agent vs crew vs deterministic workflow على نفس business process؛ compare cost, error propagation, review burden, robustness.

MOUSSAIF adoption
نأخذ separation of process from persona: HyperLoop هو workflow typed، HyperAgent worker قابل للاستبدال، والrouter يقرر إن كان agent ضرورياً أصلاً.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

Society كـauthority topology

عدد agents ليس metric نضج. topology الأنسب قد تكون peer swarm، manager→specialists، producer→verifier، أو blackboard. القرار يتعلق بكمية المعلومات القابلة للتوازي، coupling بين subtasks، تكلفة الدمج وخطر تضارب السلطة. في meta-governor، MUG لا يكتب كل شيء؛ يحدد من يملك أي claim/action ومدة التفويض.

```
Spawn if E[ParallelGain + InformationGain + VerificationGain] > CoordinationCost + MergeRisk + ContextCost
```

كل delegation يحمل lease منتهي الصلاحية وعودة «done structured: findings, evidence refs, artifacts, confidence, unresolved gaps.» من subagent لا يغلق المهمة؛ Closure Court يراجع global invariants.

صلة بحثية: Process layer + explicit state

Flows مفيدة عندما تكون الأعمال business processes ذات state واضح. لكن agent لا يجب أن يشغل node deterministic يمكن تنفيذه function. Router يختار LLM فقط عندما توجد ambiguity/reasoning، وبذلك نحافظ على reliability والكلفة.

ترجمة إلى **MUT/MUG:** MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

19 · NVIDIA NeMo Agent Toolkit: طبقة قياس وتحسين فوق أي Framework

Framework-agnostic profiling · evaluation · optimizer · APP · finetuning

NeMo Agent Toolkit لا ينافس بقية harnesses فقط، بل يضيف instrumentation/optimization فوق frameworks مختلفة. يقدّم profiler يسجل tokens/time/ calls، تحليل offline replay، forecasting، bottlenecks/concurrency؛ evals لمسار agent وال groundedness/relevance؛ optimizer لل prompts/parameters؛ Agent Performance Primitives للتوازي speculation/priority routing؛ و finetuning/RL hooks. [W19]

هذا هو أقوى مصدر لنقطة نفتقدها غالباً: **harness يجب أن يكون measurable from inside**. MOUSSAIFXT يحتاج Performance Plane مستقل: traces معيارية، success attribution، cost by node، failure taxonomy، tool latency، token/cache metrics، call tree، و counterfactual replay.

> **From MCD to measured routing**
>
> MCD لا يبقى heuristic يدوي. coefficients تتعلم offline من traces: أي features توقعت الحاجة إلى tool إضافي؟ أي routing قلل failure؟ ثم تقترح Evolution Lab أوزاناً جديدة، ولا تُرقّى إلا بعد held-out eval.

كذلك APP يفتح فكرة speculative branches: يمكن تشغيل مسارين محدودين فقط عندما يكون disagreement المتوقع عالي القيمة، بدلاً من parallelism دائم. QMCD يختار السؤال/branch الذي يفصل بين النظريات.

تشريح معماري متقدم

الفرضية الهندسية: طبقة framework-agnostic لل profiling/evaluation/optimization تجعل performance engineering مستقلة عن agent framework، مع traces وتنبؤات cost/concurrency. [W19]

موضع الحالة: Agent Performance Primitives تسمح branches/speculation/priority routing، بينما profiler يكشف tokens/time/LLM calls/bottlenecks. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: optimization قد يحسن latency/cost على benchmark ويضر semantics أو safety؛ metrics سهلة القياس قد تطغى على world-effect correctness. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier	MOUSSAIF adoption
Pareto experiments: latency/cost vs verified success؛ كل optimization يحتاج semantic regression suite وrisk budget ثابت.	Evaluation Lab يفصل SLOs إلى quality, evidence, risk, latency, token/cost, recovery، ولا يسمح optimization إذا سقط Court verdict.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: AgentOps + optimization under semantic constraints

profiling/evaluation هو ما يحول harness engineering إلى discipline قابلة للتحسين. نضيف شرطاً: optimization objective متعدد لا latency فقط، أي speculative branch أو caching أو model routing يجب أن يحافظ على evidence completeness وrisk budget.

ترجمة إلى **MUT/MUG**: MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

20 · smolagents: البساطة baseline لا بديل عنها

CodeAgent · ToolCallingAgent · MultiStepAgent · managed agents · sandbox

smolagents من Hugging Face يعرض harness صغيراً: CodeAgent، MultiStepAgent الذي يكتب tool calls في planning، Python، ToolCallingAgent JSON، callbacks، managed agents، intervals، final-answer checks، مع إمكان sandboxes. قيمته البحثية أنه يمنحنا baseline يمنع التضخم المعماري. [W20]

أي إضافة subagent society، memory tier، MCD router، HyperAbility، Court - MOUSSAIF - يجب أن تجيب: **هل تتفوق على loop صغير جيد التصميم؟** إذا لم تفعل، فهي overhead أو feature لمجال ضيق يجب تقييدها.

Honorable durability reference: Pydantic AI

ليس ضمن العشرين هنا، لكنه يقدم مثالاً قوياً على durable execution عبر Temporal/DBOS/Prefect/Restate، مع resume بعد failures/restarts وHITL. نستخدمه كمرجع للـRecovery Plane لا كبديل كامل للهارنس. [W21]

هذه الصفحة تختم atlas بمبدأ: complexity must earn its keep. هدف AL RAZZI v2 ليس جمع 20 implementations، بل تقليلها إلى primitives قابلة للقياس والاختيار.

تشريح معماري متقدم

الفرضية الهندسية: نواة صغيرة MultiStepAgent/CodeAgent/ToolCallingAgent تذكّرنا بأن baseline البسيط ضروري لفصل القيمة الحقيقية للطبقات المتقدمة. [W20]

موضع الحالة: code-as-action يسمح بدمج خطوات متعددة داخل برنامج صغير؛ managed agents تقدم delegation دون runtime ثقيل. هذا التفصيل مهم لأن failure recovery لا يمكن أن يعتمد على reasoning غير محفوظ؛ ما يهم هو state قابلة لإعادة البناء مع provenance.

حد القوة / failure mode: بساطة loop لا تعطي durability/governance/recovery تلقائياً، لكن تجاهلها يجعلنا ننسب التحسن إلى architecture معقدة ربما لا نحتاجها. ومن منظور MCD، أي طبقة لا تغير القرار أو evidence أو recovery تحت اختبار حذف تصبح overhead.

Experiment / falsifier

Ablation tournament لكل courts+ → paging+ → memory+ → layer: minimal → society+؛ نحتفظ فقط بالتحسين السببي المتكرر.

MOUSSAIF adoption

نحتفظ بـMinimalHarness كخط أساس دائم: لا يدخل primitive جديد MOUSSAIFXT إلا إذا أثبت delta فوق هذا baseline على task family مناسبة.

المعيار الحاكم: لا نرث branding أو UX؛ نرث primitive سببية ونقيسها. النجاح يجب أن يظهر في World-Effect receipt أو transfer/recovery metric، لا في جودة الشرح وحدها.

صلة بحثية: Minimal baseline + Occam court

baseline الصغير ضروري فلسفياً وتجريبياً. كلما أضفنا memory/courts/society، يجب أن نثبت أن الزيادة ليست ritual. Occam Court يقارن variant المعقد بالminimal عند نفس model/environment؛ إذا كانت gains غير مستقرة نحذف الطبقة.

ترجمة إلى **MUT/MUG**: MUT يحتفظ بالتمثيلات/البدائل القابلة للمقارنة، بينما MUG يقرر أي معيار/سلطة/driver يحكم الانتقال إلى الفعل. لذلك لا يصبح أي harness «العقل الأعلى»؛ هو substrate أو سياسة قابلة للعزل ضمن mesh.

ما وراء الأنظمة: فصل «الدماغ» عن «اليدين»

Brain/Hands separation as the core portability invariant

من Prime إلى Anthropic وOpenHands وCursor يظهر invariant مشترك: model reasoning لا ينبغي أن يملك التفاصيل الكاملة للـhost. Prime يفصل kernel-facing abstraction عن Anthropic يطرح host lifecycle، OpenHands يفصل brain/hand separation عبر Action/Observation وEventStream، وCursor يضع hands في VM. النتيجة: **نستطيع تغيير model أو execution backend دون كسر mission semantics.**



نقترح أربعة interfaces مستقلة: Reasoner يولد decisions/tool intents، ContextProvider يمَاد working set، Executor يطبق action داخل substrate؛ Verifier يعيد readback ويصدر verdict. يمكن أن يكون Reasoner Claude وExecutor OpenHands sandbox وContextProvider HyperKB وVerifier model مختلف.

الفائدة العلمية هي attribution. إذا فشل النظام، نحدد: verification failure، execution failure، context failure، decision failure أم policy failure. من دون الفصل، كل فشل يختزل إلى «النموذج أخطأ» وتضيع إمكانية التطوير.

> **Moussaif invariant**
>
> External harnesses = drivers، لا مصدر canonical authority. MUG وMission Ledger يملكان intent والحدود؛ WorldEffect/Evidence يملكان closure؛ driver يستطيع اقتراح، لا أن يعيد تعريف الهدف بصمت.

Brain/Hands ليس فصل process فقط

الفصل الأقوى هو فصل **decision substrate** عن **effect substrate.** الـbrain يقرأ state/events ويصدر intents typed؛ الـhands تنفذ فقط ما يمر عبر ToolGateway/AuthorityGraph. بهذا يمكن استبدال model أو harness دون أن يمتلك مباشرة credentials أو shell الكامل. كل effect يعود كـObservation + Receipt، لا كسطر نصي يطلب منا التصديق.

النتيجة المعمارية: MOUSSAIFXT يصبح control plane؛ AL RAZZI native driver أو Claude/Codex/Hermes تصبح execution brains/drivers، بينما local/VM/browser/cloud تصبح substrates. التبديل يتم بناءً على MCD، التكلفة، المخاطر، latency ودليل النجاح السابق.

```
Intent → Authority Check → Driver → Substrate → Effect → Receipt → Independent Verify → State Commit
```

السياق بعد نافذة النموذج: Virtual Context Memory Hierarchy

Model window as L0/L1 cache · MCAS as address space

حالة الفن converges على Prime: offload/paging يتعامل مع context كمتغيرات داخل REPL، Anthropic يجعل session log قابلاً للاستجواب، Deep Agents ينقل tool results إلى Aider، filesystem يضغط repository إلى Letta، map يوزع الذاكرة بين blocks وMemFS. هذا يعطي أساساً أقوى لـMCAS من مجرد RAG.

Tier	المحتوى	سياسة الدخول
L0	current goal + active constraints	always resident
L1	working evidence + active code slices	MCD high
L2	session/event log + summaries	paged by event range
L3	HyperKB/graph/repo map/memory	semantic/structural retrieval
L4	files/artifacts/datasets/traces	pointer + compute
L5	cloud/remote agents/external sources	authorized fetch

نعرّف Context Utility Density = decision-relevant information / resident tokens وContext Fault Rate. النظام الجيد ليس الذي يحمّل أكبر context، بل الذي يحافظ على أقل working set كافٍ ويستطيع استعادة المصدر عندما يتغير السؤال.

PEK Compiler v2

يبني Prompt Embedded Knowledge capsule من skills + selected evidence handles + authoritative constraints + active difference + goal. كل capsule يسجل سبب freshness، source، inclusion، وeviction policy.

Cognitive Paging كبديل عن myth of infinite context

نعرّف سبع طبقات: L0 current turn، L1 working set، L2 structured mission state، L3 symbol/repo map، L4 episodic trace، L5 HyperKB semantic/ graph index، L6 cold artifacts/raw files. الـmodel لا «يعرف» L6؛ يمتلك pointers وسياسة page-out. page-in لا يعني حذفاً: يحفظ digest/provenance/why- evicted كي يمكن reconstruction.

السياسة المقترحة لا تعتمد similarity وحدها. MCAS يجمع dependency centrality، recency، evidence need، authority relevance، active difference، وretrieval cost. بذلك قد يسترجع assertion قديم لأنه dependency حرج ولو لم يشبه السؤال لغوياً.

```
PageScore = α·Δ + β·EvidenceNeed + γ·Dependency + δ·Recency + ε·Transfer - λ·TokenCost
```

الذاكرة والمهارة: من التخزين إلى Capability Genome

Agent Skills · HyperSkills · progressive disclosure · provenance

Agent Skills أصبح معياراً عملياً عبر عدة أدوات، بينما Hermes/Letta/Gemini تبني skill discovery/creation. لكن «skill» قد تعني instruction folder أو procedure scripts أو. نحتاج ontology أدق: **HyperSkill** أصغر إجراء قابل للاختبار؛ **HyperAbility** composition يحقق capability أعلى؛ **HyperLoop** temporal/conditional orchestration؛ **HyperAgent** runtime role يملك authority محددة.

```
CapabilityGenome = {
 id, version, purpose, triggers, inputs, outputs,
 knowledge_refs, tools, permissions, tests, evaluator,
 provenance, failure_modes, cost_profile, promotion_state
}
```

Genome يسمح exporters إلى Agent Skills أو Gemini extension أو Claude skill، لكنه يحتفظ بعناصر لا يوفرها standard: evaluator، evidence duty، compatibility، authority، lineage. وبذلك نصنع portability دون فقد sovereignty.

Skill promotion protocol

→ experience → candidate → static/security validation → replay on source task → held-out tasks → regression suite → Court → canary → promoted. أي failure يعيد skill إلى candidate مع lineage لا يمحو التاريخ.

هذا يجسد «knowledge as OS» داخلياً: المعرفة النافعة ليست bytes؛ هي معرفة يمكن استدعاؤها، تشغيلها، اختبارها، نقلها وإبطالها. [M2-M3]

الجينوم = lineage قابلة للاختبار

كل capability تحمل: context، id/version، parent lineage، source/provenance، purpose، preconditions، input/output schema، tool/secret scopes، requirements، cost profile، tests، failure signatures، compatibility، owner، expiry، promotion state. HyperSkill هي procedure دقيقة؛ HyperAbility composition أعلى؛ HyperLoop ترتيب زمني؛ HyperAgent worker profile؛ كلها تشترك في lineage.

التطور لا يتم بالنسخ المباشر. candidate يدخل Shadow، ثم Sandbox Eval، ثم Regression Court، ثم Canary، ثم Stable. أي فشل يعيد النسخة إلى quarantine مع أثر يشرح لماذا. بهذه البنية يصبح «evolution under evidence» لا self-evolving» self-modifying code غير المحكوم.

المجتمع الوكيلي: topology قبل عدد الوكلاء

Subagent isolation · agent teams · manager-specialists · A2A peers

العشرين نظاماً تكشف أربع topologies: (1) subagent كأداة معزولة ترجع result فقط؛ (2) fork يرث context؛ (3) team له sessions مستقلة وshared tasks؛ (4) remote A2A peers عبر شبكة. لكل topology تكلفة مختلفة في coordination، context leakage، وverification.

Topology	متى تفيد	الخطر
Scout subagent	بحث كثيف/قراءة	lossy summary
Fork	حل بديل بنفس السياق	تكرار bias نفسه
Manager + specialists	مشروع decomposable	bottleneck عند manager
Peer/A2A	خدمات مستقلة/بعيدة	authority & provenance drift

نظرية Bounded Agent Society تقترح أن agent ليس identity لغوية بل tuple: role, mission_slice, context_scope, tools, authority_lease, budget, evidence_duty, stop_condition. لا يحق لل agent أن يخلق agent آخر بسلطة أكبر من سلطته، ما يشبه capability security.

Delegation Gate

Spawn إذا كان E[Δinformation + Δparallel_time] > E[coordination + context + verification + cost]. وإلا ينفذ الأصل المهمة. بذلك يصبح multi-agent optimization قراراً measurable.

Society كـauthority topology

عدد agents ليس metric نضج. topology الأنسب قد تكون peer swarm، manager→specialists، producer→verifier، أو blackboard. القرار يتعلق بكمية المعلومات القابلة للتوازي، coupling بين subtasks، تكلفة الدمج وخطر تضارب السلطة. في meta-governor، MUG لا يكتب كل شيء؛ يحدد من يملك أي claim/action ومدة التفويض.

```
Spawn if E[ParallelGain + InformationGain + VerificationGain] > CoordinationCost + MergeRisk + ContextCost
```

كل delegation يحمل lease منتهي الصلاحية وعودة «done: findings, evidence refs, artifacts, confidence, unresolved gaps.» من subagent لا يغلق المهمة؛ Closure Court يراجع global invariants.

طبقة البروتوكولات: MCP · A2A · ACP · AG-UI · Agent Skills

A polyglot protocol mesh around a sovereign internal state

النظام الحديث لا يحتاج بروتوكولاً واحداً لكل شيء. MCP يربط agent بالأدوات والبيانات؛ A2A يربط agentب agent عبر حدود framework/vendor؛ ACP يربط editor/clientب coding agent؛ AG-UI يحدد event stream بين backend وواجهة المستخدم؛ Agent Skills يحزم expertise إجرائية. في 2026 انتقل MCP إلى core stateless أكثر مع extensions/tasks/auth hardening، بينما A2A يرسخ horizontal orchestration. [P01-P05]

Layer	Protocol	MOUSSAIFXT mapping
Tools/Data	MCP	ToolGateway adapters
Agent↔Agent	A2A	Remote HyperAgent bridge
Editor↔Agent	ACP	CLI/IDE client adapter
UI↔Agent	AG-UI	Desktop event surface
Procedures	Agent Skills	HyperSkill exporter/importer

المبدأ: البروتوكولات لا تملك canonical state. جميع الرسائل تُترجم إلى MoussaifEvent داخلي typed، مع source/provenance/authority. إذا تغير protocol خارجي لا تنهار semantics الأساسية.

ProtocolBridge Court

أي adapter جديد يختبر task status، artifacts، identity، error classes، streaming، cancellation، permissions: round-trip semantics. نجاح serialization وحده غير كافٍ.

Protocol Polyglot Mesh

MCP يحل tool/resource boundary؛ A2A يحل agent discovery/delegation عبر vendors؛ ACP يربط editor/client بوكلاء coding؛ AG-UI يرسل event/state للواجهة؛ Agent Skills تحزم procedure/resources. لا يوجد بروتوكول واحد يمثل mission truth. لذلك ProtocolBridge يجب أن يحولها إلى Canonical Event Model داخلي بدل تسريب semantics كل بروتوكول إلى kernel.

الفشل الأخطر هو confused-deputy: agent عبر A2A يطلب MCP tool بصلاحيات ليست له، أو UI event يُعامل كـauthority. الحل: identity/authority تنتقل كـclaims موقعة/موثقة داخل envelope، ولا يستنتج ToolGateway السلطة من مجرد وصول الرسالة.

الأمن: containment أفضل من endless approvals

Filesystem + network isolation · trust boundaries · authority leases

مع tool-using agents يصبح prompt injection مسألة تنفيذ لا نص. Anthropic تشدد على أن filesystem isolation بلا network isolation والعكس غير كافيين؛ Gemini/OpenHands/Cursor تستخدم sandboxes/VMs بدرجات مختلفة؛ AgentDojo يبين أن untrusted tool data يمكن أن يحول model إلى منفذ لهدف مهاجم. [R12,W02,W06,W07]

نقترح Security Plane من خمس طبقات: (1) data trust labels، (2) tool capability restrictions، (3) execution sandbox، (4) network egress policy، (5) world-effect authorization/readback. وتضاف Prompt Injection Taint: النص القادم من web/email/repo غير موثوق لا يملك authority لتغيير system policy حتى لو بدا كتعليمات.

Verify

بعد action عالي الأثر، لا تعتمد tool success؛ اقرأ state من قناة مستقلة أو API readback.

Contain

قلل blast radius افتراضياً. credentials خارج scoped proxy، sandbox، read-only where possible.

Security Court يملك veto مستقل. كما أن auto-approval يُبنى من policy + classifier + scope، لا من fatigue. المخاطر لا تُعالج بإيقاف كل autonomy؛ بل بتحديد مساحة حرية آمنة يمكن للوكيل أن يعمل داخلها بسرعة.

Security model: capability security لا prompt security

كل input خارجي - صفحة ويب، message، tool observation، README، issue من agent آخر - يدخل كـuntrusted data حتى يمر classification. instructions لا تكتسب authority لأنها مكتوبة بصيغة أمر. secrets تُعطى leases قصيرة ومحددة domain/tool؛ filesystem وnetwork isolation منفصلان؛ destructive effects تحتاج reversible plan أو explicit elevation.

نضيف Prompt-Injection Court مستلهماً من AgentDojo: يحقن تعليمات مهاجمة في observations ثم يفحص هل انتقلت إلى authority plane. الاختبار يقيس attack success، utility retention، false block، وsecret exfiltration attempts، وليس فقط «هل رفض النموذج؟».

Durable Cognition: ماذا يحدث إذا قُتل الوكيل في منتصف الفعل؟

Checkpoint · journal · idempotency · compensating actions

مع المهام التي تمتد ساعات، crash ليس استثناء. Microsoft checkpoints وPydantic durable runtimes وPrime durable sessions وLetta persistent state كلها تؤكد أن الاستئناف primitive أساسية. نميز ثلاث حالات: **reasoning state**، **execution state**، **world state**. حفظ الأولين لا يثبت الثالث.

لذلك Recovery Journal يسجل قبل كل action: intent، idempotency key، expected effect، pre-state hash، authorization؛ وبعدها: post-tool response، state readback، effect receipt، rollback/compensation handle. عند restart يفحص unresolved intents بدلاً من إعادة تنفيذها عمياناً.

```
if intent_logged and no_receipt:
    state = independent_readback()
    if effect_observed: synthesize_receipt()
    elif idempotent: retry()
    else: escalate_or_compensate()
```

هذا مهم للبريد، الدفع، deployment. exactly-once ليست دائماً متاحة؛ لكن at-most-once intent + readback + compensation يمكن أن يكون contract واضحاً. الاختبار: kill process عشوائياً في 100 موضع ومقارنة duplicate/omitted effects.

الاستمرارية كمشكلة distributed systems

الوكيل الطويل يتعرض لـmodel swap، tool timeout، network partition، process restart، rate limit، crash، وhuman pause. لذلك كل transition يجب أن يحدد هل هو replay-safe. قبل effect غير idempotent نسجل intent/idempotency key؛ بعده نثبت receipt؛ بعدها فقط commit state. إذا مات النظام بين effect وcommit نستعلم العالم قبل إعادة التنفيذ.

```
RecoveryCorrect = StateRehydrated ∧ NoDuplicateEffect ∧ DependenciesValid ∧ EvidenceLinked
```

هذا يقرب Harness من workflow engine وقاعدة بيانات أكثر من chatbot. checkpoints وحدها غير كافية إذا لم تحفظ pending effects وcompensation.

Observability: من trace جميل إلى علم سببي للهارنس

Tokens · latency · call trees · bottlenecks · failure attribution

NVIDIA تجعل profiler/eval طبقة أولى؛ Harbor يوحد تشغيل agents في environments؛ OpenHands event stream وPrime accounting يوفران مسارات قابلة للتحليل. نستنتج أن MOUSSAIFXT يحتاج trace schema موحداً لا يعتمد framework.

Metric family	أمثلة
Capability	task success, partial progress, false success
Context	resident tokens, pages fetched, cache hit, rot signals
Tools	calls, retries, silent failures, information gain
Agents	spawn count, overlap, coordination cost, disagreement
Recovery	restart success, repeated effects, rollback quality
Economy	tokens, wall time, tool cost, cloud compute

الأهم **failure taxonomy**: verifier miss، execution failure، tool misuse، hallucinated state، context omission، wrong goal interpretation، routing error، stale skill، permission deadlock. كل failure يُسند إلى layer، وإلا لا يمكن تطوير النظام.

Counterfactual Replay

استعمل نفس stored observations وأعد تشغيل decision layer بسياسة مختلفة حين يكون ذلك ممكناً. هذا يقلل تكلفة ablations ويفصل بعض تأثيرات model/harness دون إعادة world actions.

من telemetry إلى causal observability

نسجل spans: mission, plan node, model call, retrieval page-in/out, tool intent, authority decision, execution, verifier, memory write, recovery. كل span يحمل evidence refs، risk class، model/driver/substrate، input/output hashes، time/cost/tokens، parent وfailure code. الهدف أن نستطيع سؤال «لماذا فشل؟» وليس فقط عرض timeline.

Failure taxonomy المقترحة: epistemic, planning, context, routing, tool, environment, permission, verification, integration, recovery, security. TraceMiner يجمع clusters ويحولها إلى candidates لا patches مباشرة.

التقييم I: هندسة البرمجيات والـTerminal

SWE-bench · SWE-Lancer · Terminal-Bench 2.0 · Harbor

SWE-bench يختبر حل issues حقيقية من GitHub داخل repos، وVerified يحدد subset قابل للحل بشهادة مهندسين. SWE-Lancer يضيف أكثر من 1,400 مهمة freelance بقيمة واقعية تصل إجمالاً إلى مليون دولار مع end-to-end grading. Terminal-Bench 2.0 ينتقل إلى مهام terminal/container واسعة ويستخدم Harbor كـofficial harness لتشغيل agents متعددة على نفس البيئة. [B01-B03]

هذه benchmarks تسمح لنا ببناء Moussaif Harness League لكن بشرط matched conditions: نفس model/version، نفس container image، نفس token/time budget، نفس network policy، نفس retries. نكرر run عدة مرات لأن stochasticity عالية. لا نقارن score منشوراً من infra مختلفة مباشرة.

المقاييس التي نضيفها

ليس pass/fail فقط: false closure، rollback count، tokens، tool calls، unnecessary edits، regressions introduced، first-valid-patch time، recovery after injected crash، evidence completeness.

Harbor مهم لأنه يفصل benchmark environment عن agent implementation، وهي نفس الفلسفة التي نريدها: AL RAZZI يصبح agent يمكن تركيبه في harness تقييم محايدة، وليس benchmark داخلياً يصمم لنفسه.

HarnessBench-MX / Software track

نجمع SWE-Lancer، SWE-bench Verified، وTerminal-Bench/Harbor في ثلاث طبقات: terminal/environment، economic realism، patch correctness. نضيف hidden mutation tests وfault injection حتى لا ينجح agent بحفظ نمط benchmark. لكل task نسجل verified، first-pass success، wall time، token/cost، duplicate effects، repair count، success، وhuman interventions.

القاعدة: scoreboard لا يساوي AGI. هذا track يقيس قدرة harness على هندسة software تحت بيئات محددة. العمومية تُختبر لاحقاً بنقل نفس primitives إلى web/research/computer-use دون إعادة تصميم hand-coded لكل benchmark.

التقييم II: الوكيل العام والحاسوب الحقيقي

GAIA · OSWorld 2.0 · WebArena · BrowserGym · τ²-bench

GAIA يقيس أسئلة يومية تتطلب reasoning, multimodality, browsing/tools. WebArena يبني مواقع وظيفية ويقيس functional correctness بدلاً من «يبدو أنجز». OSWorld نقل الاختبار إلى أنظمة تشغيل حقيقية، و2.0 OSWorld في 2026 يركز على workflows طويلة واقعية، hidden state وstreaming/dynamic environments؛ لا تزال النماذج المتقدمة بعيدة عن إغلاق نسبة كبيرة من هذه المهام. BrowserGym يعالج تجزؤ web benchmarks، وτ²-bench يختبر agents تحت policies وأدوات ومستخدم محاكى. [B04-B08]

هذه المجموعة أكثر صلة بهدف AGI-oriented harness من coding benchmark وحده لأنها تختبر **cross-modal state tracking** + **user interaction** + **policy compliance**. نقترح test suite داخل Windows: ملفات local app، terminal، settings، browser، Office، ثم tasks تتطلب عبور أكثر من application مع constraints hidden في artifact سابق.

> **World-Effect Closure**
>
> لا يحتسب النجاح من final text. يجب أن يقيس grader الحالة: file موجود وبالمحتوى الصحيح، setting تغير، form submitted مرة واحدة، app state متوافق، مع عدم وجود side effects غير مطلوبة.

هذا هو المكان الذي يجب أن تظهر فيه قيمة MUG: agent قوي بلا policy discipline قد ينجز target مع أضرار جانبية؛ score متعدد الأبعاد يجب أن يعاقب ذلك.

Generalist / Computer-use track

GAIA تختبر reasoning + tools + web؛ WebArena/BrowserGym تختبران navigation/action في مواقع واقعية؛ OSWorld 2.0 يرفع الأفق إلى workflows حاسوبية طويلة. نحتاج grader وظيفي يقرأ حالة التطبيق/الملفات لا screenshots فقط. كما نضيف perturbations: نافذة تتحرك، session تنتهي، login يعاد، UI label يتغير.

الهدف هو فصل perception failure من planning/tool failure. كل خطوة تحمل action rationale، grounding confidence مختصر، expected postcondition؛ إذا فشل postcondition يعاد التخطيط بدل الاستمرار على state متخيلة.

التقييم III: البحث، التصفح، الاقتصاد، والأفق الزمني

PaperBench · BrowseComp · TheAgentCompany · METR time horizons

PaperBench يطلب replication لـ20 ورقة ICML مع 8,316 rubrics جزئية؛ BrowseComp يختبر العثور على معلومات يصعب إيجادها عبر تصفح عميق؛ TheAgentCompany يحاكي أعمالاً مهنية عبر web/code/comms؛ METR يقيس task-completion time horizon بوحدة زمن الإنسان الخبير. [B09-B12]

هذه الاختبارات تعالج أربعة أشكال من العمومية: **research reproduction**، workplace composition، persistence in search، والاستمرار عبر تعقيد يزداد. نضيف معياراً خامساً خاصاً بنا: cross-domain transfer without harness rewrite. نفس kernel يجب أن ينتقل من code إلى research إلى business operations بتغيير HyperAbility/Tools لا إعادة تصميم core.

Research benchmark for MUT/MUG

أعط النظام ورقة، repo وdataset. المطلوب: map claims→methods→experiments، إعادة تنفيذ subset، اكتشاف اختلاف، ثم إنتاج evidence-ranked report. النجاح يقاس بالartifact/experiment لا بالتلخيص.

METR تذكر أيضاً أن time horizon ليس مدة بقاء agent متصلاً حرفياً بل human-equivalent task duration عند نسبة نجاح معينة. يجب الحفاظ على هذا التمييز كي لا نستعمل الرقم كادعاء مبالغ عن «أيام من الاستقلالية».

Research, economics, horizon

PaperBench يقيس replication research؛ BrowseComp retrieval شديد الصعوبة؛ TheAgentCompany يمزج web/code/comms؛ METR time horizon يقيس طول المهمة البشرية الذي يمكن للنماذج إتمامه باحتمال محدد. نستخدمها مع stress tests داخليّة لتمييز coordination، evidence quality، وbreadth وsurvival across sessions.

المقياس المركزي المقترح ليس مدة التشغيل بل **verified progress per unit resource**. agent قد يعمل 20 ساعة وهو يدور. Zenith يلفت إلى premature stopping؛ لذلك Gap Court يقيس الفرق المتبقي عن original spec كل جولة، وStop Court يحتاج evidence أن الفجوة تحت threshold لا مجرد self-report.

Reflexion · Voyager · SWE-smith · Continual: التعلم الذاتي الواقعي Harness

Failure as data · skill libraries · synthetic tasks · harness refinement

Reflexion يثبت فكرة أن feedback اللغوي + episodic memory يمكن أن يحسن السلوك دون weight updates. Voyager يضيف skill library، auto-curriculum، وcode قابلة للتكوين، ويولد عشرات الآلاف من software tasks للتدريب على agentic engineering. Prime Continual Harness يجعل prompts/memories/skills/subagents قابلة للتعديل كحالة دائمة. [R06-R09] SWE-smith من feedback/errors. self-verification

نصوغ منها **Failure-to-Asset Pipeline**: كل failure مهم ينتج unit test أو environment case أو skill patch أو routing feature أو data candidate. لكن التعلم لا يكتب مباشرة في production. يمر من shadow eval → court → canary، وإلا قد يتحول reward hacking إلى «تعلم».

> **AVO-like evolution without uncontrolled self-modification**
>
> نسمح بتوليد variations في prompts/skills/router thresholds/workflow structure، ثم نقيّمها على held-out missions مع cost/safety. الفائز يصبح candidate release، لا تعديل حي فوري للنواة.

بهذا يصبح self-evolution software/research process مدفوعاً بالتجربة. ويمكن لاحقاً ربط أفضل trajectories بـpost-training، لكن harness improvement يظل منفصلاً عن model weight improvement لقياس كل منهما.

من verbal reflection إلى asset pipeline

Reflexion يبين فائدة feedback لغوي + episodic memory؛ Voyager يحول الخبرة إلى executable skills مع auto-curriculum؛ SWE-smith يبين قيمة توليد task data واسعة؛ Prime Continual Harness يحرر reusable state، Zenith يضيف repeated gap-finding وindependent testers. ندمجها في Asset pipeline←Failure:

trace → root-cause hypothesis → candidate memory/skill/policy/test → sandbox eval → regression → promotion.

التعلم الحقيقي يتطلب transfer. إذا حسّن candidate المهمة التي ولد منها فقط، فهو overfit. لذلك كل candidate يختبر على held-out sibling tasks وعلى negative set للتأكد أنه لا يفرض rule بلا داع.

عندما يصبح الهارنس قديماً: Harness Plasticity Theory

Scaffolding can help today and hurt tomorrow

أحد أخطر الأخطاء هو تجميد best practice في 2025 ثم استخدامها كدستور في 2027. Anthropic توثق أن context engineering تغيّر مع النماذج الجديدة حتى أمكن إزالة نسبة كبيرة من system scaffolding، وmanaged agents صممت stable interfaces لأن تفاصيل harness غير قابلة للتنبؤ مع تقدم النماذج. [R03-R05]

نقترح **Harness Plasticity Theory**: كل scaffold له نصف عمر. قيمة prompt/tool wrapper/planner تتناقص أو تتغير مع قدرة model. لذلك كل release نموذج جديد يشغل Staleness Ablation Suite: harness minus component، current harness، baseline minimal، ثم learned/adaptive variant.

إذا حدث	التفسير	القرار
component يرفع success ويخفض cost	ما زال causal	احتفظ
success ثابت وcost أعلى	over-scaffold	بسّط
success ينخفض	stale constraint	أزل/أعد التصميم
يفيد فقط مجالاً	specialized skill	lazy-load

هذا يحمي MOUSSAIF من أن يصبح corpus ضخم يفرض ضجيجاً دائماً. المعرفة/skills تُحمّل عند الحاجة، والدستور وحده يبقى صغيراً ثابتاً نسبياً.

Staleness كدين تقني معرفي

كل scaffolding يعكس limitations نموذج عند تاريخ معين. مع نماذج أقوى قد يصبح decomposition زائد، context compression ضار، أو tool schema ضيقة. Harness Plasticity تعني قدرة النظام على قياس قيمة كل primitive دورياً وحذفه أو تبسيطه عندما يصبح negative overhead.

```
NetValue(component,t)=ΔVerifiedSuccess + ΔTransfer - ΔCost - ΔLatency - ΔRisk
```

Evolution Lab يشغل quarterly ablations عبر model generations. لا نسمح للهوية المعمارية أن تتحول إلى dogma: حتى HyperSkill/MCD court نفسها تخضع لاختبار حذف/تبسيط.

Moussaif Adoption Matrix: نرث العلاقات لا الأسماء

Adopt · extend · isolate · reject

المصدر	Adopt	Extend in MOUSSAIF
Prime	persistent programmatic context	MCAS + MUG-owned canonical state
Claude	sandbox/hooks/evaluator	typed policy hooks + Courts
Codex	thin local driver	common driver API
Hermes	memory/skills/runtime portability	promotion + provenance
OpenCode	client/server + LSP	multi-client daemon + semantic tool tier
Gemini	extensions/skills/subagents/policy	Capability Genome
OpenHands	event-stream runtime	Evidence-carrying event sourcing
SWE-agent	ACI design	Tool Epistemics
Aider	repo map	Project World Model
Cline/Roo	roles/checkpoints/permissions	Authority Leases
Cursor/Manus	VM + parallelism	tiered substrate + bounded society
Deep Agents	offload/isolation	MCAS paging
Letta	git memory + self-edit tiers	Memory Ontology + Courts
MAF/CrewAI	durable explicit workflows	HyperLoop graph contracts
NVIDIA	profile/eval/optimize	Evolution Performance Plane
smolagents	minimal loop baseline	permanent anti-bloat benchmark

القاعدة النهائية: لا vendor lock-in في canonical ontology، لا direct write من external agent إلى constitutional state، ولا feature بلا ablation. هذا يحافظ على هوية MOUSSAIFXT كـmeta-harness وليس «تجميع 20 CLIs».

أربع حالات adoption

Adopt: primitive مثبت ومتماشٍ مع عقودنا مثل checkpoints أو event logs. **Adapt**: نأخذ العلاقة ونغير representation، مثل skills→HyperSkill with provenance. **Federate**: لا نعيد البناء؛ نكتب إلى driver. Claude/Codex/Hermes/Manus. **Reject/Quarantine**: feature لا يضيف delta أو يكسر /sovereignty evidence.

هذا يمنع تضخم AL RAZZI. الهدف ليس جمع 20 codebases داخل binary واحدة، بل امتلاك kernel أصغر يمكنه اكتشافها وتوجيهها ومراقبتها، بينما HyperKB يحفظ ما تعلمناه عن performance/failures لكل driver.

نظريات Harness الجديدة - المجموعة الأولى

HT2-01 … HT2-05 · mechanisms, tools, falsifiers

HT2-01 · Harness Microkernel Sovereignty

النظرية: العمومية الهندسية ترتفع عندما تبقى goal/authority/state/evidence في microkernel صغير، وتتحول harnesses الخارجية إلى drivers. **الأداة:** SubstrateRegistry. **الهدم:** إذا تفوق monolithic harness باستمرار في portability/recovery دون كلفة governance.

HT2-02 · Evidence-Carrying Action

كل action عالي الأثر يجب أن يحمل intent + authorization + before/after evidence + rollback. **الأداة:** EffectReceiptLedger. **المقياس:** false-success/duplicate-effect rate.

HT2-03 · Cognitive Paging

context الفعال memory hierarchy؛ retrieval هو page fault resolution لا مجرد similarity search. **الأداة:** ContextPager/MCAS. **الهدم:** إذا full-context baseline يتفوق تحت نفس cost.

HT2-04 · Harness Plasticity

كل scaffold قابل للتقادم مع تقدم model. **الأداة:** HarnessAblator. **البروتوكول:** staleness suite عند كل model release.

HT2-05 · Bounded Agent Society

multi-agent يفيد فقط عندما authority/context/budget/evidence محددة وعائد التفويض موجب. **الأداة:** AuthorityGraph + DelegationGate.

HT2-01 إلى HT2-05: من أسماء إلى فرضيات

Microkernel Sovereignty: فصل kernel عن drivers يجب أن يخفض coupling ويزيد replaceability دون خفض success. **Evidence-Carrying Action:** كل effect يحمل receipt يجب أن يخفض false-completion. **Cognitive Paging:** tiered context يجب أن يتفوق على naive long-context/summary عند corpora أكبر من window. **Harness Plasticity:** periodic ablation يجب أن يمنع scaffolding debt. **Bounded Agent Society:** authority leases يجب أن تخفض unsafe/conflicting actions مقابل free-form teams.

لكل نظرية falsifier محدد: إذا لم يظهر فرق على benchmark مختار تحت نفس model/environment، تُخفض من «نظرية تشغيلية» إلى design preference.

نظريات Harness الجديدة - المجموعة الثانية

HT2-06 ... HT2-10 · durable agency and evolution

HT2-06 · Interruption-Resilient Agency

agency طويلة الأفق تُقاس بالقدرة على resume after arbitrary interruption مع حفظ world semantics، لا بطول chat. **الأداة:** RecoveryJournal + crash injector.

HT2-07 · Failure-to-Asset Conversion

الفشل يصبح أصل تعلم فقط إذا تحول إلى reproducible case ثم improvement ينجح held-out. **الأداة:** FailureMiner + CandidateFactory.

HT2-08 · Protocol Polyglot Mesh

لا protocol واحد يغطي tools/agents/UI/editor/skills؛ kernel يحافظ على event semantics ثم adapters. **الأداة:** ProtocolBridge.

HT2-09 · Evidence-Weighted Substrate Routing

اختيار model+harness+host يجب أن يتعلم من performance registry تحت task signature، لا brand. **الأداة:** EvidenceRouter/MCD calibration.

HT2-10 · Capability Genome

skill/ability/loop/agent يجب أن يملك lineage/tests/permissions/evaluator ويمكن تصديره standards. **الأداة:** GenomeRegistry + SkillCompiler.

هذه النظريات ليست claims مثبتة. كل واحدة مربوطة بأداة واختبار يمكن أن يفشلها، وفق منهج المشروع الذي يرفض رفع المصطلح إلى نظرية ما لم يغير قراراً أو تنبؤاً أو قياساً. [M1-M3]

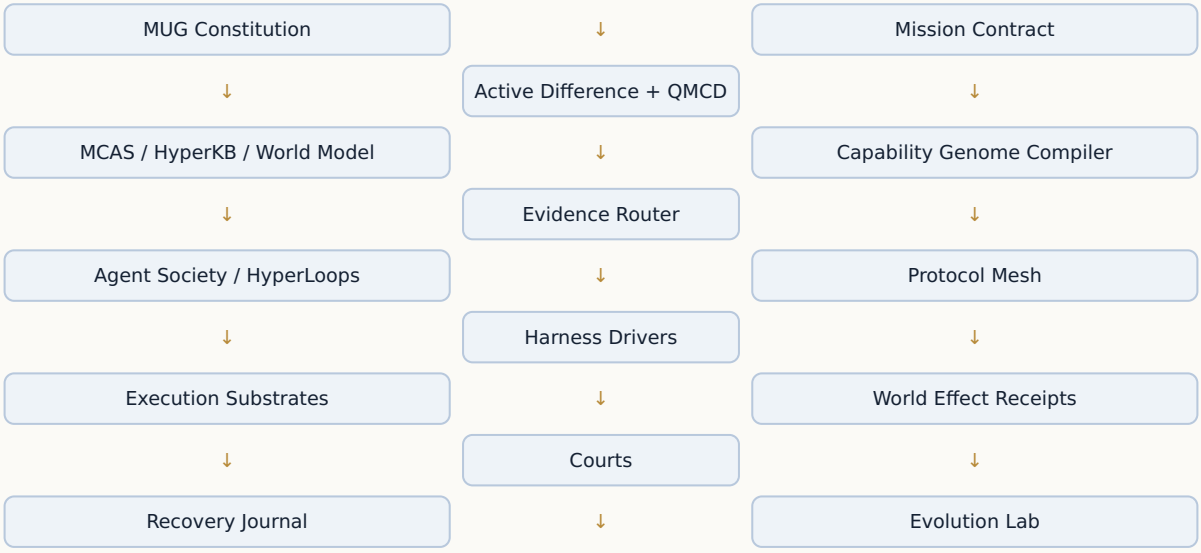
HT2-06 إلى durability :HT2-10 والتطور

Interruption-Resilient Agency: correctness بعد crash معيار قدرة. **Failure-to-Asset Conversion:** الفشل لا يصبح تعلماً إلا إذا رفع held-out transfer. **Protocol Polyglot Mesh:** canonical event model يجب أن يقلل integration-specific logic. **Evidence-Weighted Substrate Routing:** اختيار model/ driver من سجل نتائج verified substrate يجب أن يتفوق على static routing. **Capability Genome:** lineage + tests + promotion gates يجب أن تسمح evolution قابلة للrollback.

```
Promotion(candidate) iff Δheldout>θ ∧ Regression=0 ∧ Risk≤budget ∧ ProvenanceComplete
```

AL RAZZI / MOUSSAIFXT v2: المعمارية المستهدفة وبرنامج التنفيذ

14 subsystems · production gates · 120-day experimental roadmap



14 subsystems: Constitution Kernel، Mission Ledger، MCD/QMCD Router، MCAS ContextPager، HyperKB/WorldModel، GenomeRegistry /HarnessRegistry/Driver API، ProtocolBridge، AuthorityGraph، ExecutionBackend mesh، EvidenceLedger/EffectReceipt، Courts IndependentVerifier، RecoveryJournal، Performance/Evolution Plane.

0-30 يوم	31-60	61-90	91-120
typed contracts + 4 drivers + crash recovery	/MCAS paging + skill genome + protocols MCP/A2A AG-UI	+ Harbor/SWE-bench/OSWorld subset security suite	matched-budget league + ablations + publishable report

Production gate: لا نطلق «AGI harness» كادعاء قدرة. نطلق AGI-oriented governed hyper-harness فقط بعد: real build missions، restart tests، security injection suite، matched harness comparisons، false-success measurement، cross-domain missions، وتكرار E5-style داخل سلم المشروع. نجاح Architecture لا يساوي AGI، لكنه يجعل السؤال قابلاً للاختبار.

من MVP إلى experimental AGI-oriented harness

نقترح ثمانية planes منفصلة: (1) Mission/Goal; (2) Context/HyperKB/MCAS; (3) Capability Genome; (4) Agent Society; (5) Driver/Substrate Registry; (6) Authority/Security; (7) Evidence/Courts/World Effect; (8) Recovery/Evolution/Observability. كل plane يملك typed API، ولا يستدعي model مباشرة إلا عبر ModelGateway.

برنامج 120 يوماً: 0-30 تثبيت contracts + receipts + crash recovery؛ 31-60 MCAS + repo/symbol maps + drivers لـHermes/Codex/Claude/native؛ 61-90 HarnessBench-MX + AgentDojo-style injection + parallel society; 91-120 FailureMiner + candidate factory + shadow/canary promotion. معيار الخروج ليس عدد features بل E4/E5 evidence في السلم الداخلي: local test ثم repeatable comparative experiment.

I: Harnesses, Frameworks, Repositories المراجع الخارجية

Primary sources · accessed 05 Sep 2026

[W01] Prime Intellect. Prime Agent: RLM-native coding/research harness; persistent IPython, recursive subagents, Continual Harness. https://github.com/PrimeIntellect-ai/prime-agent · https://arxiv.org/abs/2608.23552

[W02] Anthropic. Claude Code / sandboxing / long-running harness engineering. https://www.anthropic.com/engineering/claude-code-sandboxing · https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents

[W03] OpenAI. Codex CLI repository. https://github.com/openai/codex

[W04] Nous Research. Hermes Agent repository and docs. https://github.com/NousResearch/hermes-agent

[W05] OpenCode. Open-source coding agent; provider agnostic, LSP, client/server. https://github.com/anomalyco/opencode

[W06] Google. Gemini CLI: extensions, Agent Skills, subagents, policy, sandbox. https://github.com/google-gemini/gemini-cli

[W07] OpenHands. Runtime architecture and event-driven agent/runtime design. https://docs.openhands.dev/openhands/usage/architecture/runtime

[W08] SWE-agent. Agent-Computer Interface design. https://github.com/SWE-agent/SWE-agent

[W09] Aider. Repository map and architect/editor modes. https://github.com/Aider-AI/aider

[W10] Cline. CLI: Plan/Act, MCP, checkpoints, subagents/teams, schedules. https://github.com/cline/cline

[W11] Roo Code. Modes, Orchestrator, custom permissions, MCP. https://github.com/RooCodeInc/Roo-Code

[W12] Continue. Source-controlled AI checks and agent configurations. https://github.com/continuedev/continue

[W13] Cursor. Cloud Agents and long-running agents research. https://cursor.com/docs/cloud-agent · https://cursor.com/blog/long-running-agents

[W14] Manus. Cloud Computer and Wide Research. https://help.manus.im/en/articles/15392111-what-is-the-cloud-computer · https://manus.im/blog/introducing-wide-research

[W15] LangChain. Deep Agents overview/context/subagents. https://docs.langchain.com/oss/python/deepagents/overview

[W16] Letta. Letta Code memory-first agent harness. https://github.com/letta-ai/letta-code

[W17] Microsoft. Agent Framework and workflow checkpoints. https://github.com/microsoft/agent-framework · https://learn.microsoft.com/en-us/agent-framework/workflows/checkpoints

[W18] CrewAI. Agents, crews and persistent Flows. https://docs.crewai.com/

[W19] NVIDIA. NeMo Agent Toolkit: profiling, evaluation, optimization. https://github.com/NVIDIA/NeMo-Agent-Toolkit

[W20] Hugging Face. smolagents / MultiStepAgent / CodeAgent. https://huggingface.co/docs/smolagents/reference/agents

[W21] Pydantic AI. Durable execution with Temporal, DBOS, Prefect, Restate. https://pydantic.dev/docs/ai/capabilities/durable_execution/overview/

[W22] Intelligent Internet (2026). *Zenith: A Continuous-Improvement Harness for Long-Running Tasks*. Cross-harness orchestration for Claude Code, Codex and Hermes via MCP/ACP; repeated gap-finding, workers/testers, replanning and stopping discipline. https://github.com/Intelligent-Internet/zenith

II: Papers, Benchmarks, Protocols, Books المراجع الخارجية

Research anchors · not a claim that any benchmark measures AGI alone

[R01] Karten et al. (2026). Prime Agent: A Self-Improving RLM Harness. arXiv:2608.23552.

[R02] Anthropic (2025). Effective harnesses for long-running agents.

[R03] Anthropic (2025/2026). Effective context engineering; Managed Agents: Decoupling the brain from the hands.

[R04] Anthropic (2026). Harness design for long-running application development / long-running primitives.

[R05] Anthropic (2026). New rules of context engineering for newer model generations.

[R06] Shinn et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. NeurIPS.

[R07] Wang et al. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models.

[R08] Yang et al./SWE-smith project (2025). Scaling Data for Software Engineering Agents. https://swesmith.com/

[R09] CoALA: Cognitive Architectures for Language Agents, TMLR; used as background for memory/action/decision decomposition.

[R12] Debenedetti et al. (2024). AgentDojo: Prompt Injection Attacks and Defenses for LLM Agents. NeurIPS D&B.

[R15] Anthropic (2026). Quantifying infrastructure noise in agentic coding evals.

[B01] SWE-bench: Can Language Models Resolve Real-World GitHub Issues? ICLR 2024; SWE-bench Verified.

[B02] OpenAI (2025). SWE-Lancer: 1,400+ real-world freelance software engineering tasks.

[B03] Terminal-Bench 2.0 + Harbor (2025-2026). https://www.harborframework.com/

[B04] Mialon et al. (2024). GAIA: A benchmark for General AI Assistants. ICLR.

[B05] Xie et al. (2024) OSWorld; Yuan et al. (2026) OSWorld 2.0.

[B06] Zhou et al. (2023). WebArena: A Realistic Web Environment for Building Autonomous Agents.

[B07] BrowserGym Ecosystem for Web Agent Research, TMLR 2025.

[B08] τ-bench / τ²-bench: tool-agent-user interaction under policies and APIs.

[B09] OpenAI (2025). PaperBench: Replicating 20 ICML research papers with 8,316 rubric tasks.

[B10] OpenAI (2025). BrowseComp: 1,266 hard-to-find browsing questions.

[B11] TheAgentCompany: consequential workplace tasks across web/code/comms.

[B12] METR (updated May 2026). Task-Completion Time Horizons of Frontier AI Models.

[B13] AgentDojo security benchmark: 97 tasks / 629 security test cases.

[P01] Model Context Protocol, 2026-07-28 specification. https://modelcontextprotocol.io/

[P02] Agent2Agent (A2A) protocol. https://a2a-protocol.org/

[P03] Agent Client Protocol (ACP). https://github.com/agentclientprotocol/agent-client-protocol

[P04] Agent User Interaction Protocol (AG-UI). https://docs.ag-ui.com/

[P05] Agent Skills open standard; Anthropic engineering note, 2025.

[W22] Intelligent Internet (2026). *Zenith: A Continuous-Improvement Harness for Long-Running Tasks*. Cross-harness orchestration for Claude Code, Codex and Hermes via MCP/ACP; repeated gap-finding, workers/testers, replanning and stopping discipline. https://github.com/Intelligent-Internet/zenith

[R16] Intelligent Internet (2026). *From RALPH to Zenith: Designing Harnesses for Long-Running Agents*. Technical report and ablation study distributed with the Zenith repository.

المراجع III: الكتب + Corpus MOUSSAIF + عقد البحث النهائي

Books · internal anchors · experimental closure

[BK01] Chip Huyen (2024). *AI Engineering: Building Applications with Foundation Models*. O'Reilly. خاصة فصول evaluation, model selection, agents, data/eval pipelines.

[BK02] Chip Huyen (2022). *Designing Machine Learning Systems*. O'Reilly. reliability, scalability, maintainability, adaptability, monitoring and iterative systems.

[BK03] Anthropic. *Building Effective AI Agents: Architecture Patterns and Implementation Frameworks*. eBook/resource.

[M1] MOUSSAIF AGI OS / MOUSSAIXT - Deep Engineering Architecture, 50-page internal report: 12-layer stack, tool contracts, World Effect, Courts, evolution and cross-environment runtime.

[M2] MOUSSAIF Unified AGI - claim-ranked research/engineering book: E0-E5 evidence ladder; MCD/QMCD/MCAI/HAKIM; HKOS; HyperSkills/Abilities/Loops/Agents; Mission DAG; verification/reconstruction.

[M3] AGI-Oriented Knowledge-Base as Cognitive OS - internal theory: HyperKB as operational substrate linking knowledge → skill → loop → memory → falsification.

عقد البحث بعد Part II

لا نعتبر هذا atlas إثباتاً لتفوق MOUSSAIFXT. ما ينتقل من literature/repositories إلى نظامنا هو **فرضيات adoption**. الإثبات يبدأ عند تشغيل drivers حقيقية على environments متطابقة، تسجيل traces، حقن failures، اختبار الأمن، ثم ablation كل primitive. النظرية التي لا تغير prediction أو decision أو measured outcome تُخفض إلى اسم هندسي أو metaphor.

المسار المقترح للنشر العلمي: (1) ورقة Harness Microkernel + Evidence-Carrying Action؛ (2) ورقة Cognitive Paging / MCAS على corpora أكبر من context؛ (3) HarnessBench-MX benchmark يفصل model/harness/environment؛ (4) دراسة recovery/security؛ (5) longitudinal study لـ-Failure-to-Asset وHarness Plasticity.

الخلاصة: أفضل ما في حالة الفن ليس «Agent واحد خارق»، بل انتقال الوكالة من prompt إلى نظام: agents، skills، sandboxes، tools، compute، context، state، recovery، evidence، protocols وevaluation. هدف MOUSSAIFXT/AL RAZZI هو جعل هذه العناصر تحت حاكمية موحدة قابلة للفصل والقياس والاستبدال، بحيث تصبح قوة النموذج مادة خام، لا نقطة انهيار النظام.