هندسة الهارنس الوكيلي للذكاء العام الموجّه

من Claude Code وPrime Agent وCodex وHermes إلى AL RAZZI / MOUSSAIFXT

معمارية توحيد الهارنسات الناجحة، عشر نظريات جديدة للهارنس، وخمس نظريات لـ Agentic AGI داخل MUT · MUG · MOUSSAIFXT

Harness Engineering · Virtual Context · HyperKnowledge · Tool Use · Computer Use · Evidence · World Effect · Self-Evolution

إعداد وتركيب بحثي: زهير مساعف / Zouhair Moussaif

نسخة بحثية هندسية · 05 سبتمبر 2026 · A4 / Amiri / Arabic RTL

تنبيه منهجي: هذه الورقة تقترح برنامجاً معمارياً وهندسياً قابلاً للاختبار، لا تعتبر وجود طبقات أو أسماء أو حجم corpus دليلاً على تحقق AGI. النجاح يُقاس بأثر العالم، الاختبارات، إعادة التشغيل، المقارنة بخطوط أساس، ونقل التعلم عبر مهام جديدة.

الملخص والأطروحة المركزية

Abstract · Research Question · Contributions

تبحث هذه الورقة في سؤال هندسي محدد: كيف يمكن بناء AGI-oriented harness لا يكتفي باستدعاء نموذج لغوي وأدوات، بل ينسق المعرفة، السياق الممتد، التنفيذ، الوكلاء، الحماية، التحقق، واستثمار الخبرة داخل نظام قابل لإعادة البناء؟ تنطلق الورقة من أن الهارنسات الناجحة أثبتت أن جودة النظام لا تتحدد بالنموذج وحده. Prime Agent يبرز قيمة البيئة البرمجية المستمرة وContinual Harness والوكلاء المتفرعين؛ Hermes Agent يبرز الذاكرة والمهارات والتشغيل متعدد البيئات؛ Codex يبرز الفصل بين sandbox والموافقات، والواجهة البرمجية والمهارات؛ بينما توجيهات Anthropic للوكلاء طويلة الأفق تؤكد حفظ الحالة، التحقق، توزيع العمل، وضبط كثافة الاستدعاءات. [U1-U7]

المساهمة هنا ليست تجميع هذه الأنظمة في Frankenframework، بل استخراج الآليات التي أثبتت نفعها ثم إعادة صياغتها كعقود داخل AL RAZZI Hyper Harness المربوط بـ MOUSSAIFXT. المشروع الداخلي يعرّف MUT كطبقة توحيد النظريات، وMUG كحاكمية البرنامج، وMOUSSAIXT كغلاف تنفيذ عابر للبيئات؛ كما يضع سلسلة MCD/QMCD/MCAI/HAKIM-MCD و→ HyperKB → HyperSkill → HyperAbility → HyperLoop → HyperAgent ومحاكم الدليل والأثر. [M1-M5]

الإسهامات الرئيسة

- معمارية إنتاجية موحدة تتعامل مع Claude/Prime/Codex/Hermes كـ execution substrates قابلة للاستبدال، لا كملاك للحالة السيادية.
- تعريف Virtual Context / MCAS بحيث يتجاوز النظام قيد نافذة النموذج عبر paging وcompute-to-context دون الادعاء بسياق لا نهائي.
- عشر نظريات Harness جديدة، لكل منها آلية، أداة، بروتوكول، فرضية قابلة للهدم ومقياس.
- خمس نظريات Agentic-AGI تربط MUT وMUG وMOUSSAIXT بالحاكمية المعرفية، أثر العالم، واستمرارية الهوية الوكيلية.
- برنامج benchmark وablation وإعادة تشغيل يميز architecture-rich عن evidence-rich.

حد الادعاء: الورقة تصوغ research program + engineering architecture؛ أي نتيجة عن AGI لا ترتفع عن فرضية إلا بعد قياس مستقل ومتكرر.

المنهج وسلّم الإثبات

Claim-ranked engineering · Evidence tiers · Falsification

تستخدم الورقة منهجاً مزدوجاً. الأول **هندسي مقارن**: نراجع الآليات الواضحة في الهارنسات الحالية ونفصل بين ما هو runtime mechanism وما هو مجرد واجهة أو اصطلاح. الثاني **تركيبي مساعفي**: نربط تلك الآليات بسلالة المشروع التي تجعل الفرق النشط والسؤال الحاكم والحاكمية والدليل وأثر العالم وحدات تشغيل. لذلك لا يكفي أن نقول إن النظام يملك memory أو skills؛ يجب تحديد متى تُقرأ، من يملكها، كيف تُختبر، وكيف تساهم في إغلاق مهمة.

الرتبة	المعنى في هذه الورقة	مثال
E0	غير متحقق / فرضية أو تسمية	نظرية جديدة بلا تجربة
E1	موصوف أو مذكور في مصدر	ميزة موثقة في upstream
E2	Artifact أو عقد هندسي موجود	Schema / registry / module
E3	تشغيل واختبار محلي موثق	Golden test / trace
E4	مقارنة بخط أساس تحت ميزانية متطابقة	Ablation / matched-budget
E5	تكرار مستقل عبر مهام وبيئات	Reproduction / cross-domain

يعكس هذا السلم مبدأ corpus الداخلي: الملف المولّد لا يساوي النشر، ونداء الأداة لا يساوي أثر العالم. [M1,M3] كما تتبنى الورقة اختبار الحذف: إذا أزلنا طبقة مثل MCD routing أو Closure Court ولم يتغير النجاح أو الكلفة أو التعافي، فإن القيمة المعمارية لهذه الطبقة في ذلك المجال تصبح غير مثبتة.

قاعدة البحث

كل مفهوم جديد يجب أن يغيّر قراراً أو يضيف مقياساً أو يولّد اختباراً. المصطلح الذي لا يبدّل سلوك النظام ولا يمكن هدمه يُخفض إلى استعارة أو وصف، لا إلى نظرية.

خريطة الهارنسات الحديثة

Successful patterns rather than brand imitation

النمط المشترك في الهارنسات القوية هو نقل الذكاء من single prompt إلى نظام تشغيل حول النموذج. تختلف المشاريع في موضع القوة: بعضُها يجعل البيئة البرمجية المستمرة مركزاً، بعضُها يجعل الذاكرة والمهارة محوراً، وبعضها يركز على أمان تنفيذ الأوامر ومراحل الموافقة. لذلك نميز خمس طبقات مقارنة: **Context، Capability، Execution، Continuity، Verification.**

الهارنس	أقوى آلية قابلة للتبني	المحدودية التي نعالجها
Prime Agent	Persistent IPython/RLM، recursive subagents، Continual Harness، daemon continuity	نحتاج فصل التطور عن السيادة وإضافة World-Effect/Courts
Claude-style agentic engineering	long-horizon state tracking، parallel tool use، subagent discipline، filesystem rediscovery	نحتاج runtime محايداً للنموذج مع سجل claims/authority
Codex	sandbox + approval separation، typed SDK، streaming، AGENTS/skills، workspace scope	نضيف cross-harness routing وknowledge-as-OS
Hermes	persistent memory، self-created skills، progressive disclosure، checkpoints، cron، multiple backends	نحتاج ترقية المهارة عبر benchmark/court بدل self-promotion
NVIDIA AVO	long-horizon variation، persistent lineage، supervisor، stagnation redirect	نحوّلها إلى Evolution Plane محكوم لا self-modification مباشر
mini-SWE / Aider	minimal baseline، repo map، Git transactionality	مقياس لمنع الإفراط في المعمارية

بالتالي، الهدف ليس أن يعمل AL RAZZI مثل كل واحد منهم في آن واحد، بل أن يملك capability negotiation يسمح باستيراد القوة المناسبة لكل عقدة في Mission DAG مع بقاء الحالة السيادية والمعرفة والحاكمية داخل Moussaif.

Prime Agent: البرمجة المستمرة بدل تضخم السياق

Persistent RLM kernel · Continual Harness · Recursive agency

يقدم Prime Agent نموذجاً مهمّاً للهارنس البرمجي: لكل جلسة persistent IPython kernel، بينما يملك المضيف TypeScript تنفيذ الأبناء والمحاسبة ودورة الحياة. تظل المتغيرات والبيانات والبرامج في البيئة الخارجية، ويمكن للنموذج التعامل معها عبر كود بدلاً من إعادة إدخالها كاملة في كل turn. كما يسجل Continual Harness ملاحظات prompt والذاكرة ووصف المهارات ومواصفات subagents كحالة قابلة للتنقيح والرجوع. [U1,U2]

نستخلص أربع قواعد. أولاً، **context is programmable**: قيمة RLM ليست Python في ذاته، بل إمكان إرسال computation إلى البيانات. ثانياً، الاستدعاء المتكرر لوكيل ابن يجب أن يكون عملية runtime قابلة للمحاسبة لا فقرة نصية. ثالثاً، الاستمرارية تتطلب daemon/session state لا الاعتماد على بقاء الطرفية. رابعاً، تحسين السلوك الدائم يجب أن ينفصل عن system prompt غير القابل للعبث.

التحويل إلى Moussaif

HyperKernel → Prime RLM، Continual Cognition Store → Continual Harness، HyperAgent → child agents، Delegation → persistent context variables، EVCF/MCAS handles. ويضاف فوق ذلك: Evidence Ledger، Authority Gate، World-Effect verification، Closure Court.

المعيار الذي نرفض عنده التبني المباشر هو أن يصبح Prime هو مالك المهمة أو الذاكرة السيادية. في AL RAZZI ينبغي أن يكون PrimeAgentAdapter substrate اختيارياً: يستقبل عقداً محدوداً ويرجع structured result وevidence وartifacts، بينما يحتفظ Moussaif بالحالة الكلية.

Claude-style agentic engineering

Long-horizon state · Subagents · Context discipline · Verification

تشدد إرشادات Anthropic للوكلاء طويلة الأفق على التقدم المتدرج، حفظ الحالة في filesystem، إعادة اكتشاف الحالة بعد نافذة سياق جديدة، استعمال subagents عندما تكون هناك فائدة من التوازي أو عزل السياق، والتفريق بين الأفعال المحلية القابلة للعكس والأفعال المشتركة أو المدمرة. كما تدعم النماذج الحديثة التوازي في tool calls وتستفيد من أدوات تحقق مثل browser/computer use في المهام الطويلة. [U4]

هذه النقاط مهمة لأن الهارنس الناجح لا يحاول منع model reset، بل يجعل reset حالة طبيعية. عندما يستطيع الوكيل بدء نافذة جديدة، قراءة progress وGit وtests واستعادة تعريف المهمة، فإن الاستمرارية تصبح خاصية runtime لا خاصية ذاكرة سرية داخل النموذج. وهذا يطابق هدف Self-Reconstruction Memory في corpus.

أربعة مبادئ نتبناها

- Filesystem rediscovery: الحالة الأساسية قابلة لإعادة البناء من artifacts وevents لا من ملخص واحد.
- Subagent economy: لا spawn إلا للتوازي أو العزل أو الاستقلالية النقدية.
- Parallel tool calls when independent: التوازي مورد اقتصادي يُقاس بالزمن والكلفة وليس سلوكاً افتراضياً دائماً.
- Reversibility-aware autonomy: الأفعال المحلية القابلة للعكس أوسع من النشر أو الإرسال أو تغييرات البنية المشتركة.

نطور هذا المنطق عبر MUG: كل subagent يحمل authority ceiling، وكل عملية توازي تمر عبر AgentPopulationController كي لا تتحول كثرة الوكلاء إلى ضجيج.

Codex: هندسة التنفيذ والعزل

Sandbox/approval separation · Typed SDK · Skills · Repository harness

يعطي Codex درساً مركزياً: sandbox يحدد ما يمكن للعملية تقنياً الوصول إليه، بينما approval policy يحدد متى يحتاج الفعل إلى موافقة. الفصل بينهما يمنع خلط القدرة بالبروتوكول. كما تعرض واجهة Python الحالية نمطاً typed لبدء thread مع read_only/workspace_write/full_access، مع streaming وinterrupt وsteering. [U5,U6]

ومن جهة هندسة repository، توصي مواد OpenAI باستخدام AGENTS.md كسياق دائم، وفصل goals/plans/build/context، وتحويل الإجراءات المتكررة إلى skills بدلاً من تكرار prompt يدوي. [U7] هذه الفكرة تلتقي مباشرة مع HyperSkill Factory، لكن Moussaif يضيف شرطاً أقوى: المهارة الجديدة لا تُعتمد لأنها مفيدة مرة، بل تمر عبر trace موثق، اختبار، مقارنة baseline ومحكمة تعميم.

Codex → Moussaif

Sandbox يصبح جزءاً من Secure Execution Plane، approval يدخل Authority Engine، AGENTS.md يدخل Project Context Genome، وskills تتحول إلى HyperSkills ذات I/O وpreconditions وevidence وrecovery. أما SDK/event streaming فيصبح مرجعاً لواجهة AL RAZZI التي يجب أن تكون صالحة لـ CLI وـ UI بدون parsing للنص.

تنبيه إنتاجي: الأمان الفعلي لا يجوز أن يعتمد على حسن نية model أو prompt. يجب أن تكون القيود تحت طبقة agentic cognition، وأن تُختبر محاولات تجاوزها عبر MCP أو remote executor أو adapters.

Hermes: الذاكرة والمهارة والتشغيل متعدد البيئات

Persistent memory · Progressive skills · Checkpoints · Cron · Backends

Hermes Agent هو المرجع الأكثر قرباً من بيئتنا المحلية: ذاكرة مستمرة منسقة، skills تُحمّل حسب الحاجة، إنشاء مهارات من الخبرة، بحث في الجلسات السابقة، checkpoints قبل تعديل الملفات، subagents معزولون، MCP، cron، وterminal backends تشمل local وDocker وSSH وبيئات serverless. [U3] كما يكتشف ملفات سياق المشروع مثل AGENTS.md وCLAUDE.md ويستعمل progressive disclosure لتقليل token load.

القيمة المعمارية لهيرمس هي أنه يبرهن عملياً أن الوكيل يمكن أن يعيش خارج جلسة chat واحدة وأن يجمع بين CLI والبوابات والمهارات والذاكرة. لكن في Moussaif نريد درجة أعلى من الفصل: الذاكرة الحوارية ليست HyperKnowledge تلقائياً، والمهارة التي كتبها الوكيل لا تصبح production capability تلقائياً.

نضيف

- Epistemic status
- Skill promotion court
- World-effect verification
- Cross-harness canonical mission state
- EVCF virtual addressing
- Runtime performance registry

نحتفظ

- Context discovery
- Persistent curated memory
- Skills as procedural assets
- Checkpoints/rollback
- Multiple terminal backends
- Cron/background execution

وبالتالي، HermesAdapter داخل MOUSSAIFXT لا يختزل Hermes إلى shell. يمكنه التفاوض على capabilities مثل memory، terminal، MCP، browser، skills، وsubagents، لكن AL RAZZI يظل مالك MissionContract وEvidence وWorldEffect.

المصفوفة التركيبية: ماذا نرث وماذا نطور؟

Engineering inheritance matrix

المحور	Prime	Claude	Codex	Hermes	Moussaif upgrade
السياق	RLM/IPython	multiwindow state	repo instructions	context files	EVCF + MCAS + PEK
الذاكرة	harness state	filesystem continuity	project artifacts	curated memory	typed memory tiers + epistemic ledger
المهارات	executable Python	subagent/tool discipline	skills	self-created skills	HyperSkill + Generalization Court
الوكلاء	recursive/A2A	native orchestration	threads/tasks	isolated subagents	HyperAgent + population economics
الأمان	host constraints	reversibility guidance	sandbox + approvals	approval /command container	MUG Authority + Secure Execution Plane
التطور	refine/	self-correction	skills/postmortems	skills improve	shadow → eval → court → canary
الإغلاق	goals/gates	verify state	tests/diff	checkpoints	WorldEffect + ClosureCourt

هذه المصفوفة تنتج مبدأ **الوراثة الوظيفية**: لا نرث اسم feature، بل نرث علاقة سببية. فإذا كان persistent kernel يحسن معالجة السياق الكبير لأنه يحتفظ بالمتغيرات خارج prompt، فالمطلوب هو المحافظة على تلك العلاقة حتى لو كان implementation مختلفاً. وإذا كان checkpoint قبل edit يقلل خسارة العمل، فالمطلوب عقد rollback موحد عبر hosts.

قاعدة عدم التضخم

أي طبقة جديدة يجب أن تتفوق في ablation على baseline أبسط. وجود mini-SWE كوكيل صغير وفعال يذكّرنا أن الكفاءة قد تأتي من loop جيد وأدوات واضحة، وأن المعمارية الثقيلة يجب أن تبرر overheadها. [U8,U9]

AL RAZZI الحالي: من Specification إلى Runtime

Verified internal baseline and remaining gap

وفق تقرير النشر الداخلي الحالي، توجد طبقات فعلية في AL RAZZI OS: Command/Event contracts، runtime مفصول عن CLI، session persistence في project discovery، SQLite وregistries، PEK compiler، HyperKB indexing، repo map للـLoop/Ability/HyperSkill/ Agent، Zenith-style Active Difference، MCD router، delegation، Windows subprocess sandbox، Git transaction ToolGateway، Evidence Ledger، أربع independent verifier وCourts، Self-Reconstruction candidates، إضافة إلى golden suites. [M0]

لكن الورقة لا تساوي تقرير التنفيذ بالدليل النهائي. لذلك نضع V0 hypothesis: **AL RAZZI هو baseline instrumentable وليس بعدُ برهاناً على تفوق harness.** المطلوب الآن تحويل المكونات إلى مسار إنتاجي قابل للمقارنة: provider حقيقي، tool effects حقيقية، browser/computer tasks، context restart mid-mission، cross-harness delegation أكبر من نافذة النموذج، ثم ablation تحت ميزانية متساوية.

الفجوة الإنتاجية

- إثبات أن كل صندوق معماري يقع على causal path حقيقي داخل mission.
- قياس false-success بدلاً من الاكتفاء بعدد الاختبارات الناجحة.
- قياس context efficiency وrecovery وcross-domain transfer.
- إثبات أن self-evolution يرفع أداء held-out tasks ولا يخرق governance.
- إثبات أن MOUSSAIXT يستطيع تبديل substrate دون فقد الحالة أو المعنى.

هذه الفجوة هي نقطة انطلاق النظريات الجديدة في الصفحات التالية: النظريات ليست بديلاً عن الهندسة، بل تعريفاً لما يجب أن تبرهنه الهندسة.

MOUSSAIXT · MUG · MUT: البنية الثلاثية

Theory Unification · Program Governance · Execution Shell

تحدد وثائق المشروع ثلاث بنيات متكاملة: MUT يوحد النظريات والعلاقات والأسئلة والأدوات ومرشحي التجارب، MUG يحافظ على اتساق قواعد البحث والتشغيل والحاكمية، وMOUSSAIXT يربط models/tools/memory/agents/verification بالبيئات المحلية والسحابية. [M2] هذا الفصل مهم لأنه يمنع ثلاث أخطاء: أن تتحول النظرية إلى runtime غير مراقب، أو تتحول الحاكمية إلى شعارات خارج الكود، أو يصبح host الخارجي هو صاحب القرار.

MOUSSAIXT

السؤال: أين وكيف تُنفذ العقدة؟ أي model/harness/tool/host يحققها بأفضل evidence/cost؟

MUG

السؤال: من يملك سلطة تغيير الحالة، ما حدود الاستقلال، وما دليل الإغلاق؟

MUT

السؤال: ما الذي نعتقد أنه صحيح، وما العلاقات بين النظريات، وما التجربة التي تفصل بينها؟

تقترح هذه الورقة أن AL RAZZI هو **Harness Bridge** بين الثلاثة. عند كل mission، يستدعي MUT المعرفة والنظرية الملائمة، يفرض MUG السلطة والمحاكم، ويستعمل MOUSSAIXT substrate التنفيذ. لا يجوز لأي واحد منها أن يبتلع الاثنين الآخرين.

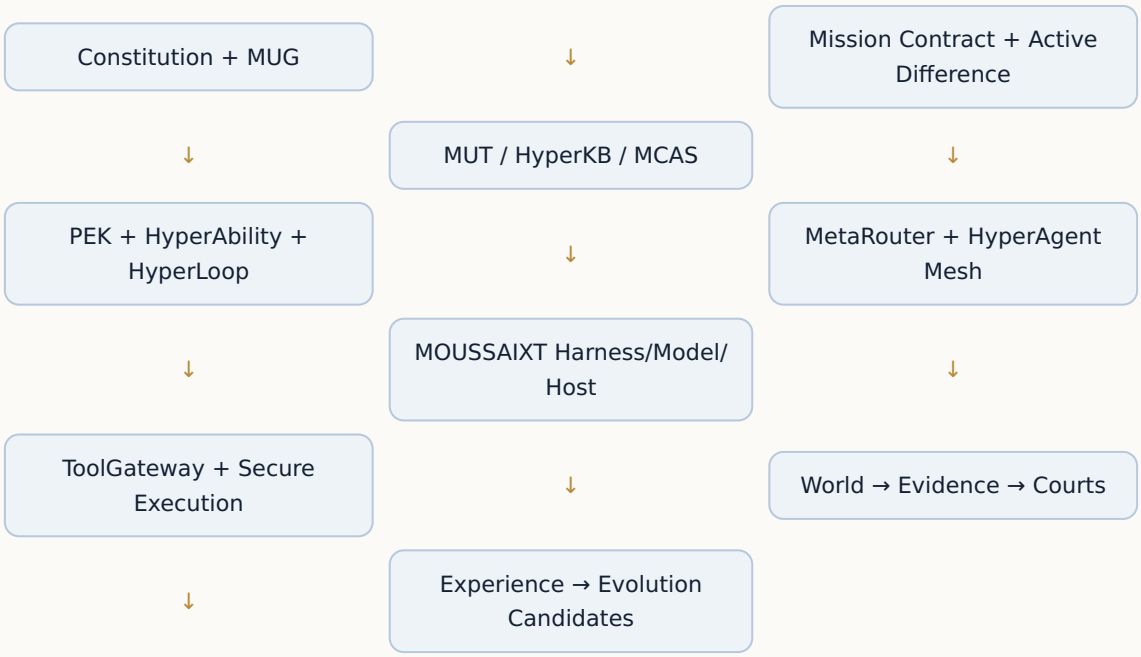
العقد الكلي

Goal → Governing Question → Active Difference → MUT knowledge → MUG authority → PEK/HyperAbility → Mission DAG → MOUSSAIXT substrate → Tool/Computer World → Evidence → Courts → Memory/Evolution

المعمارية الكلية المقترحة

From 12-layer stack to production-grade harness

يحافظ التصميم على السلم الداخلي الموثق: Constitution/Safety، Goal/Question/Difference، MCD/QMCD/MCAI، HKOS/World Model، Retrieval، HyperSkills، HyperAbilities، HyperLoops، Agents/Router، Tools/Execution، Verification/World Effect Memory/Evolution. [M1] ويضيف V5/V6 ثلاث planes عرضية: Virtual Context Plane، Secure Execution Plane، وEvolution Plane.



المبدأ الجديد هو أن **الطبقات ليست pipeline جامداً**. التحقق قد يعيد فتح world observation، Active Difference قد يعدّل World Model، وContextFault قد يستدعي resource إضافياً دون إعادة mission. لذلك runtime الأنسب هو event-driven state machine مع DAG، لا سلسلة prompts ثابتة.

Universal Provider & Harness Fabric

Models and harnesses as replaceable cognitive substrates

لكي يكون النظام فعلاً agentic OS يجب ألا يكون Claude-only أو GPT-only. تُفصل ثلاثة قرارات: **Model selection**، **Harness selection**، **Host selection**. فقد يعمل Hermes فوق نموذج محلي، أو AL RAZZI native فوق API بعيد، أو Prime Agent كبيئة RLM لعقدة بحثية، أو Codex لعقدة software engineering.

Capability Manifest

كل Model/Harness/Host يعلن persistence، context، structured output، tools، vision، coding، reasoning: capabilities، estimated cost/latency، historical reliability، checkpoint، streaming، sandbox، browser/computer، subagents.

يقوم MetaRouter بحل عقدة المهمة إلى requirement vector ثم يختار substrate بالاعتماد على fit وreliability وprivacy وcost وlatency وtool support وcontext need. لا يُعطى أي brand أولوية ثابتة. وبعد التنفيذ تُسجل النتيجة في RuntimePerformanceRegistry لتصبح routing policy evidence-driven.

هذا يخلق فرقاً جوهرياً عن multi-provider gateways العادية: gateway يختار endpoint، بينما Harness Fabric يختار شكل *cognition* والتنفيذ أيضاً. ويمكن لعقدتين في DAG نفسها استعمال harnessين مختلفين، بشرط أن يبقى protocol للنتائج والأدلة موحداً.

السياق الافتراضي الممتد: EVCF / MCAS

Virtual context as address space, not inflated prompt

أحد أكبر حدود الهارنسات هو مساواة قدرة النظام بنافذة النموذج. نقترح Extensible Virtual Context Fabric (EVCF) الذي تطور إلى Moussaif Cognitive Address Space (MCAS). الفكرة ليست سياقاً لا نهائياً، بل أن model context يصبح **resident working set**، بينما تبقى الملفات والـHyperKB والجداول والرموز والـtraces والصور والحالة الوكيلية موارد معنونة خارجية.

كل مورد يأخذ URI مثل ctx://evidence/... , cog://hypothesis/... , ctx://kb/... , ctx://repo/.... ويمكن للـruntime تنفيذ search/ page/query/compute/pin/evict/prefetch/diff. إذا احتاج reasoning مورداً غير resident يحدث ContextFault، فيحمّل الحاكم فقط الجزء اللازم ويعيد PEK.

معادلة تشغيلية

C_eff = quality(materialization, retrieval, compute, continuity) ،C_eff ≠ C_model. نعرّف القدرة السياقية الفعلية تقريبياً: تحت قيد نافذة C_model. زيادة bytes الخارجية وحدها لا تزيد الذكاء؛ الذي يزيده هو الوصول الصحيح منخفض الكلفة إلى المورد الصحيح.

يُختبر EVCF على repository أو corpus أكبر من نافذة النموذج: النجاح يتطلب حل المهمة دون إدخال كامل corpus، مع تسجيل pages accessed، tokens avoided، resident tokens، cache hit، ومعدل context faults.

Computer Use وTool Calling كطبقة أثر

From function call to governed world transition

الهارنس agentic لا يكتمل بـ function calling فقط. كل provider قد يعبّر عن tool call بطريقة مختلفة، لذلك يحول AL RAZZI أي طلب إلى ToolIntent موحد: expected effect، authority، agent، mission، arguments، tool_id. ثم يمر عبر ToolGateway قبل التنفيذ. النتيجة تصبح ToolObservation مع structured result وartifacts وevidence وerror وworld-effect candidate.

في computer use نطبق هرمية الموثوقية: API أولاً، ثم CLI، ثم direct data/filesystem، ثم DOM، ثم accessibility/UI automation، ثم visual clicking. المرئي هو fallback عندما توجد واجهة بنيوية أدق. كل ComputerAction يحمل intent وtarget وexpected effect وrisk وverification method.

After Action

· Re-observe · compare expected/observed delta · capture evidence · update World Model

Before Action

· Observe state · classify effect · check authority · checkpoint if needed

القاعدة الحاسمة: **click issued ≠ task succeeded.** نجاح computer use لا يثبت إلا عندما يطابق observed state الأثر المتوقع أو تقرر المحكمة أن الفرق المتبقي مقبول.

الأمن والحاكمية: القدرة لا تعني الإذن

Authority, sandbox, trust and reversible effects

نتبنى الفصل الذي يتضح في Codex ونعيده داخل MUG: Sandbox Policy هو قيد تقني، أما Authority Policy فهو قرار معنوي/تشغيلي بشأن السماح. يمكن أن يملك process وصولاً تقنياً لملف، لكن المهمة غير مخولة بحذفه. كما لا يجوز اعتبار نجاح sandbox approval دليلاً على أن المستخدم قصد نشر أو إرسال شيء.

نقترح أربع trust dimensions مستقلة: ProjectTrust، PluginTrust، ToolTrust، RemoteHostTrust. محتوى repository أو webpage يعامل كبيانات غير موثوقة ولا يستطيع تغيير الدستور أو authority. الأسرار تدخل عبر CredentialBroker كمراجع scoped ولا تُنسخ في PEK أو logs.

Effect classes

READ_ONLY → REVERSIBLE_WRITE → COMPENSATABLE_WRITE → IRREVERSIBLE/EXTERNAL_EFFECT. ترتفع متطلبات checkpoint/evidence/approval مع صعوبة التراجع ونطاق الضرر.

يقترح البحث اختباراً adversarial واضحاً: توفير MCP tool يمكنه remote execution ثم طلب عملية منعها sandbox المحلي. يجب أن يمنع MUG انتقال السلطة عبر حدود adapter دون إعادة تصنيف الأثر والثقة. هذه ليست feature إضافية بل اختبار لسلامة النظام السيادي.

Evidence · Courts · World Effect

Completion is an observed state transition

تضع السلالة الداخلية مبدأ صارماً: output النموذج لا يساوي أثر العالم. [M1,M3] لذلك تمثل كل مهمة بثلاث حالات: before، intended delta، observed after. تُربط الأدلة بـ claims محددة: process result، browser state، database query، API readback، test result، diff، artifact hash أو human approval.

تعمل Courts كطبقات منفصلة لا كـ"critique prompt" واحد. Evidence Court يسأل هل الادعاء مسنود؛ Falsification Court يبحث عن اعتراض حاسم؛ Regression Court يحمي وظائف سابقة؛ Closure Court يقارن Mission Contract بالواقع؛ وNo-Loss Court يحمي الأصول غير المقصودة.

معيار الإغلاق

COMPLETE ⇔ mandatory criteria satisfied ∧ no critical remaining gap ∧ evidence sufficient ∧ world effect verified ∧ no unacceptable regression. وإلا تكون الحالة PARTIAL أو BLOCKED أو REOPEN.

التجديد هنا هو ربط هذه المحاكم بـMUG لا بنموذج بعينه. يمكن أن يكون المنتج Claude أو Codex أو Hermes، لكن التحقق النهائي يظل مستقلاً عن brand وعن confidence المولد.

التطور طويل الأفق: من AVO إلى Moussaif Evolution Plane

Variation, lineage, stagnation and governed promotion

عرضت NVIDIA في أغسطس 2026 Agentic Variation Operators (AVO) كمعمارية طويلة الأفق تجمع persistent memory مع supervision وcandidate lineage، وذكرت تشغيل بحث تحسين kernels لسبعة أيام واستكشاف مئات الاتجاهات، ثم نقل النظام إلى ARC-AGI-3 وتحقيق نتيجة كاملة على public set. [U10] نستخدم هذه النتيجة كدليل على أهمية harness design، لا كبرهان عام على AGI.

نحوّل المبدأ إلى MoussaifEvolutionPlane: كل variation ينتج candidate له parents وhypothesis وmutation type وscore وevidence. عندما ينخفض ProgressSlope أو يتكرر الخطأ، يفعّل StagnationDetector بدائل: model جديد، harness مختلف، context slice آخر، hypothesis جديدة، أو تبسيط.

حد التطور

candidate لا يغير production مباشرة. المسار: PROPOSED → SHADOW → BENCHMARK → COURT → CANARY → PROMOTED/REJECTED. تغييرات الأمن والدستور لا تُرقى ذاتياً.

هذه الطبقة تجعل التعلم من الخبرة قابلاً للاختبار: التحسن الحقيقي هو أن النسخة الجديدة تتفوق على frozen baseline في held-out tasks دون زيادة غير مضبوطة في الكلفة أو regression.

النظرية H1: الغشاء الحاسوبي السيادي

Sovereign Harness Membrane Theory

H01 · Sovereign Harness Membrane Theory

تقترح النظرية أن الهارنس ليس غلافاً خارجياً للنموذج، بل غشاءً ينظم التبادل بين cognition والعالم. النموذج يولد اقتراحات، لكن الغشاء يحدد ما يدخل من سياق، وما يخرج كفعل، وما يُقبل كذاكرة. لذلك تتحول جودة الوكيل من خاصية داخل النموذج إلى خاصية مشتركة بين model وharness وworld coupling.

الآلية / Mechanism

Agency = f(Model Capability, Context Governance, Action Boundary, Evidence Feedback). إذا ضعف أي عامل قد يكون النموذج قوياً لكن الوكالة هشة. الغشاء يملك adapters وauthority وcontext وevidence ولا يسمح بانتقال غير موثق بين الداخل والخارج.

البروتوكول التشغيلي

Observe → classify ingress/egress → authorize → execute → readback → ledger. كل crossing يصبح event يمكن تتبعه.

الأدوات المقترحة

- HarnessMembraneInspector
- AuthorityBoundaryTracer
- ContextIngressPolicy
- EffectEgressPolicy

قابلية الهدم والقياس

Falsification. إذا لم يرفع الغشاء reliability أو يقلل false-success مقارنةً بحل مباشر بنفس model/tools، فالدعوى تفشل في ذلك المجال.
Metrics. context efficiency، recovery rate، unauthorized-effect rate، false-success rate.

صلة النظرية بالبنية: MUG يملك السيادة؛ MOUSSAIXT يوفّر العبور؛ AL RAZZI يطبق الغشاء كـruntime contract.

النظرية H2: سيادة السياق الافتراضي

Virtual Context Sovereignty Theory

H02 · Virtual Context Sovereignty Theory

الذكاء الوكيلي طويلِ الأفق لا ينبغي أن يكون رهينة لما هو resident في نافذة النموذج. السيادة السياقية تعني أن النظام، لا النموذج، يملك قرار ما يجب الاحتفاظ به، وما يُعنون خارجياً، وما يُسترجع أو يُحسب عند الحاجة.

الآلية / Mechanism

Cognitive Working Set = selected materialization(MCAS, mission state, Δ, model budget). السياق الخارجي يبقى addressable، والـPEK هو materialization مؤقت لا نسخة من المعرفة الكلية.

البروتوكول التشغيلي

عند الحاجة إلى معلومة غير resident: emit ContextNeed → resolve URI → choose representation → materialize minimal page → continue.

الأدوات المقترحة

- VirtualContextGovernor
- ContextFaultResolver
- ResourcePager
- ComputeToContextEngine
- ContextTrace

قابلية الهدم والقياس

Falsification. تفشل النظرية إذا لم يتمكن النظام من حل مهام على corpus أكبر من نافذة model دون full stuffing، أو إذا كانت أخطاء paging أعلى من baseline مباشر.

Metrics. mission success، cache hit، page-fault latency، tokens avoided، useful-context ratio.

صلة النظرية بالبنية: امتداد مباشر لـEVCF/MCAS وHyperKB وROFSC ضمن MUT.

النظرية H3: تحويل المعرفة إلى قدرة تنفيذية

Executable Knowledge Transduction Theory

H03 · Executable Knowledge Transduction Theory

لا تصبح المعرفة ذات قيمة agentic لأنها قابلة للاسترجاع فقط. المعرفة تنتقل من claim إلى قدرة عندما تُربط بشروط تفعيل، أدوات، إجراءات، أدلة وفشل. النظرية تصف التحويل من HyperKnowledge إلى HyperSkill ثم HyperAbility/HyperLoop.

الآلية / Mechanism

K → Contract → Skill → Composition → Verified Effect → Experience. كل مرحلة تضيف executable semantics دون حذف provenance.

البروتوكول التشغيلي

استخرج trace ناجحاً → حدد invariant الإجرائي → أنشئ candidate skill → اختبره على حالات جديدة → سجل الحدود → روّجه أو ارفضه.

الأدوات المقترحة

- KnowledgeUnitCompiler
- HyperSkillFactory
- CapabilityLinker
- GeneralizationCourt

قابلية الهدم والقياس

Falsification. إذا لم يرفع تحويل المعرفة reuse أو transfer مقارنة باسترجاع النص الأصلي، فالنظرية تضعف.
Metrics. skill transfer gain، repeated-task cost، regression، scope precision.

صلة النظرية بالبنية: تجسيد مباشر لـKnowledge-as-OS في [M4] corpus وربطه بالهارنس.

النظرية H4: الحوسبة الموزونة بالفرق

Difference-Weighted Compute Theory

H04 · Difference-Weighted Compute Theory

المشكلة ليست ندرة compute فقط بل سوء توزيعه. ينبغي تخصيص التفكير والاسترجاع والوكلاء والتحقق بحسب الفرق الذي يمكنه تغيير القرار، لا بحسب حجم النص أو طول المهمة.

الآلية / Mechanism

MCD/QMCD .Budget_i ∝ w(Δ_i) = impact × uncertainty × dependency × risk × information_gain / cost
يمثلان آلية تقدير الفرق والسؤال الحاكم.

البروتوكول التشغيلي

استخرج ΔA → احسب الوزن → خصص /mode FAST
STANDARD/DEEP/COURT → أعد الوزن بعد كل evidence
.event

الأدوات المقترحة

- ActiveDifferenceExtractor
- MCDScorer
- ComputeAllocator
- QuestionRouter

قابلية الهدم والقياس

Falsification. اختبار الحذف: إذا أعطى توزيع متساوٍ للcompute نفس النجاح والكلفة، فلا قيمة إضافية للموجه الوعائي.
Metrics. success under fixed budget، unnecessary tool/model calls، latency، cost per verified delta.

صلة النظرية بالبنية: امتداد هندسي لMCD/QMCD وComputational Vascularity.

النظرية H5: اتحاد الهارنسات واستقلال الركيزة

Harness Federation & Substrate Independence Theory

H05 · Harness Federation & Substrate Independence Theory

إذا كانت المهمة والسيادة والذاكرة مرتبطة بهارنس واحد، فإن النظام لا يكون عاماً بل تابعاً للركيزة. تقترح النظرية أن كل harness خارجي يجب أن يكون capability provider محدوداً تحت Mission Contract موحد.

الآلية / Mechanism

Canonical Mission State ⊥ Execution Substrate. يمكن تبديل Claude/Codex/Hermes/Prime أو local harness بين العقد دون تبديل هوية المهمة.

البروتوكول التشغيلي

حل → node requirements → query available harnesses → route → collect typed result/evidence → merge only verified artifacts into canonical state.

الأدوات المقترحة

- HarnessAdapter
- HarnessRouter
- CapabilityNegotiator
- CrossHarnessResultNormalizer

قابلية الهدم والقياس

Falsification. تفشل النظرية إذا أدى تبديل substrate إلى فقد حالة، اختلاف semantics غير مكتشف، أو عدم قدرة النظام على إكمال نفس contract.

Metrics. normalized evidence completeness، substrate failover success، routing gain، cross-harness continuity.

صلة النظرية بالبنية: هي قلب MOUSSAIXT باعتباره execution shell عابراً للبيئات.

النظرية H6: إغلاق الفعل المقترن بالدليل

Evidence-Coupled Action Closure Theory

H06 · Evidence-Coupled Action Closure Theory

الهارنس القوي لا يغلق المهمة عند اكتمال generation أو tool return. الإغلاق نفسه عملية معرفية تربط claim بأثر خارجي وباختبار لمطابقة delta المطلوبة.

الآلية / Mechanism

Closure = match(IntendedDelta, ObservedDelta, Evidence, Constraints). evidence لا يكون ملحقاً بعد التنفيذ بل جزءاً من contract قبل التنفيذ.

البروتوكول التشغيلي

قبل الفعل حدد expected evidence → نفذ → اقرأ الحالة → التقط evidence → شغل close/reopen court.

الأدوات المقترحة

- EvidenceLedger
- WorldEffectVerifier
- ClosureCourt
- ArtifactHasher

قابلية الهدم والقياس

Falsification. تُهدم النظرية إذا لم تقلل false-success على مهام الأدوات مقارنةً بإغلاق مبني على self-report.

Metrics. verified completion، court reopen rate، readback coverage، false-success.

صلة النظرية بالبنية: تجسيد لـWorld-Effect Loop ومحاكم الفجوة/التكذيب في MOUSSAIF.

النظرية H7: التنويع المحفوز بالركود

Stagnation-Triggered Variation Theory

H07 · Stagnation-Triggered Variation Theory

الاستقلال الطويل لا يحتاج loop أطول فقط، يحتاج اكتشاف أن الاستراتيجية الحالية لم تعد تنتج فرقاً مفيداً. عند الركود يجب تغيير الفرضية أو الأداة أو model/harness أو تمثيل السياق.

الآلية / Mechanism

If d(Progress)/dt < ε for window W → trigger VariationOperator. التغيير نفسه موجه بسجل الفشل وليس random mutation.

البروتوكول التشغيلي

راقب ProgressVector → اكتشف plateau → ولّد alternative candidate مع hypothesis → اختبر cheapest discriminator → retain/reject.

الأدوات المقترحة

- StagnationDetector
- AgenticVariationOperator
- CandidateLineageGraph
- MetaHarnessSupervisor

قابلية الهدم والقياس

Falsification. إذا زاد variation الكلفة دون رفع escape-from-stagnation أو objective score، يجب تعطيله.

Metrics. compute wasted before redirect، progress slope، candidate diversity، stagnation recovery.

صلة النظرية بالبنية: يربط AVO بالMCD: الركود نفسه يصبح Active Difference عالي الأولوية.

النظرية H8: الاستمرارية بإعادة البناء

Reconstructive Continuity Theory

H08 · Reconstructive Continuity Theory

الوكيل طويل الأفق لا يحتاج الحفاظ على نفس hidden context؛ يحتاج القدرة على إعادة بناء الحالة الحاسمة بعد restart أو context reset. الاستمرارية خاصية للآثار والعقود، لا لذاكرة نموذجية مستمرة.

الآلية / Mechanism

Identity_t+1 = reconstruct(MissionContract, event log, artifacts, checkpoints, unresolved gaps). الملخص وحده غير كافٍ إذا أخفى الفشل أو السلطة أو side effects.

البروتوكول التشغيلي

عند restart: load snapshot → replay tail → verify external state fingerprints → rebuild PEK → resume READY node.

الأدوات المقترحة

- StateReconstructor
- CheckpointManager
- EventJournal
- ContinuityScanner

قابلية الهدم والقياس

Falsification. يفشل الادعاء إذا استمر agent نصياً لكنه يعيد نفس الأخطاء، يكرر effects، أو ينسى constraints بعد restart.

Metrics. recovery time، state divergence، duplicate-effect rate، restart continuity.

صلة النظرية بالبنية: امتداد لـSelf-Reconstruction Memory وHermes/Prime daemon continuity.

النظرية H9: أيض القدرة

Capability Metabolism Theory

H09 · Capability Metabolism Theory

المهارات ليست assets ثابتة؛ يجب أن تدخل دورة حياة تشبه الأيض: اكتساب، اختبار، استعمال، تقوية، تراجع، حجر أو إحلال. بذلك يتجنب النظام تراكم آلاف skills متشابهة أو مكسورة.

الآلية / Mechanism

Experience → Candidate → Validation → Active Capability → Monitoring → Repair/Quarantine/Supersede.

الجودة تاريخية ومجالية.

البروتوكول التشغيلي

بعد كل invocation حدث success/failure/cost/scope → إذا هبطت reliability تحت threshold فعّل quarantine أو repair candidate.

الأدوات المقترحة

- CapabilityHealthRegistry
- SkillDeduplicator
- QuarantineManager
- GeneralizationCourt

قابلية الهدم والقياس

Falsification. تُهدم النظرية إذا لم تقلل skill bloat أو routing errors أو regression مقارنة بسجل ثابت غير مراقب.

Metrics. transfer success، quarantine recovery، duplicate rate، active-skill precision.

صلة النظرية بالبنية: تطوير لـHyperSkill Factory وzero-duplication governance.

النظرية H10: جينوم الهارنس التكيفي

Adaptive Harness Genome Theory

H10 · Adaptive Harness Genome Theory

يمكن تمثيل configuration الكلي للهارنس كجينوم من سياسات السياق والتوجيه والوكلاء والأدوات والتحقق والذاكرة. التعديل يصبح نسخة candidate قابلة للمقارنة، لا تغييرات مبعثرة في ملفات config.

الآلية / Mechanism

Genome = {context, router, models, harnesses, skills, loops, verification, recovery, security}. كل genome يحمل parent وbenchmark signature.

البروتوكول التشغيلي

freeze production genome → generate candidate → shadow replay → held-out benchmark → court → canary → promote/rollback.

الأدوات المقترحة

- HarnessGenomeRegistry
- ShadowRunner
- TrajectoryReplayEngine
- CanaryPromoter
- AutoRollback

قابلية الهدم والقياس

Falsification. تفشل النظرية إذا لم يمكن إرجاع الفرق في الأداء إلى genome version أو إذا أدى التطور إلى regressions غير مكتشفة.
Metrics. reproducibility، canary rollback، regression rate، gain over frozen genome.

صلة النظرية بالبنية: الجسر بين Self-Reconstruction وEvolution Laboratory وAVO-style lineage.

تركيب النظريات العشر

A unified Harness Science hypothesis

تتكامل النظريات العشر في مسار واحد: H1 يعرّف الهارنس كغشاء سيادي، H2 يحرره من نافذة model، H3 يحول المعرفة إلى capability، H4 يوزع compute على الفرق، H5 يحرره من substrate، H6 يربط الإغلاق بالأثر، H7 يعطيه قدرة تغيير الاستراتيجية، H8 يحفظ الهوية عبر restart، H9 يمنع تراكم قدرات ميتة، H10 يجعل configuration نفسه قابلاً للتطور المقاس.



الفرضية الجامعة هي أن **العمومية الوكيلية لا تنتج فقط من general model، بل من قدرة harness على إدارة انتقالات المعرفة والحالة والفعل عبر الزمن والبيئات.** هذه فرضية هندسية قابلة للهدم: يمكن مقارنة full stack مع minimal agent loop على نفس model وtools وbudget. إذا لم يظهر gain في transfer، long-horizon completion أو recovery، فالتعقيد غير مبرر.

أداة جامعة: Harness Causal Profiler

يبني graph يربط كل طبقة بـ events والقرارات والنتائج، ثم يتيح ablation: ماذا يحدث إذا أوقفنا ContextGovernor أو MCD أو Court أو skill factory؟ الهدف تحويل architecture claims إلى causal evidence.

نظرية A1: العمومية عبر أثر العالم المحكوم

Governed World-Effect Generality

H01 · Governed World-Effect Generality

تقترح النظرية أن معياراً أقوى للAgentic AGI هو القدرة على تحقيق أهداف متنوعة في بيئات جديدة مع الحفاظ على حاكمية الدليل والسلطة والتراجع، لا مجرد حل benchmark نصي. العمومية هنا علاقة بين adaptation وverified world effect.

الآلية / Mechanism

Generality ≈ Transfer × VerifiedEffect × Governance × Recovery. أي مكوّن صفري يخفض الادعاء حتى لو كانت الإجابات اللغوية قوية.

البروتوكول التشغيلي

ثبت model/budget → شغل مهام /software/research → قس transfer وverified effects وrecovery → browser/data → قارن baselines.

الأدوات المقترحة

- GeneralityBenchmarkRunner
- WorldEffectSuite
- CrossDomainTransferMatrix

قابلية الهدم والقياس

Falsification. إذا تفوق النظام فقط في مجال واحد أو بسبب budget أكبر، لا تدعم النتائج عمومية الوكالة.

Metrics. policy adherence، recovery under novelty، verified completion، cross-domain transfer.

صلة النظرية بالبنية: يجسد هدف MOUSSAIF كـAGI-oriented architecture دون ادعاء AGI مثبت.

نظرية A2: الإغلاق المزدوج MUT–MUG

MUT–MUG Dual Closure Theory

H02 · MUT–MUG Dual Closure Theory

المهمة العامة تحتاج إغلاقين مستقلين: إغلاق معرفي يقرر ما إذا كانت النظرية/الخطة متماسكة ومسنودة، وإغلاق حاكمي يقرر ما إذا كان الفعل مسموحاً ومتوافقاً مع الهدف والسلطة. MUT وMUG يمثلان هذين القطبين.

الآلية / Mechanism

Accept(Action) = EpistemicClosure_MUT ∧ GovernanceClosure_MUG. يمكن أن تكون الخطة صحيحة تقنياً لكنها غير مخولة، أو مخولة لكنها معرفياً ضعيفة.

البروتوكول التشغيلي

قبل high-impact node: حل claims/evidence في MUT → حل authority/constraints في MUG → لا تنفيذ إلا إذا اتفق الإغلاقان.

الأدوات المقترحة

- MUTClaimResolver
- MUGAuthorityResolver
- DualClosureGate

قابلية الهدم والقياس

Falsification. تُهدم النظرية إذا لم يقل هذا الفصل أخطاء scope أو false certainty أو unauthorized effects.

Metrics. repair after disagreement، authority violation، epistemic rejection precision.

صلة النظرية بالبنية: تجسيد مباشر لتعريف النظام الثلاثي في [M2].

نظرية A3: الوكالة المستقلة عن الركيزة

MOUSSAIXT Substrate-Independent Agency

H03 · MOUSSAIXT Substrate-Independent Agency

إذا كانت هوية الوكيل مرتبطة بنموذج أو provider أو حاسوب واحد، فلا توجد وكالة نظامية مستقرة. تقترح النظرية أن الوكالة تُعرّف بالحالة والأهداف والمعرفة والسلطة، بينما models/harnesses/hosts أعضاء قابلة للاستبدال.

الآلية / Mechanism

Agent Identity = Mission + WorldModel + Memory + Authority + Capability Graph، لا current model = session.

البروتوكول التشغيلي

ابدأ mission على checkpoint → substrate A → فشل A → استأنف على B → تحقق من semantics وeffects.

الأدوات المقترحة

- MOUSSAIXTAdapterMesh
- StatePortableMission
- HarnessFailoverTester

قابلية الهدم والقياس

Falsification. يفشل الادعاء إذا كانت النتائج تعتمد اعتماداً غير معلن على context مخفي أو API خاص لا يمكن ترجمته.

Metrics. provider portability، semantic drift، state preservation، failover success.

صلة النظرية بالبنية: هذه النظرية تعطي MOUSSAIXT دوراً أعمق من adapter collection: حفظ agency عبر substrata.

نظرية A4: الهوية الوكيلية بإعادة البناء الذاتي

Self-Reconstructive Agentic Identity Theory

H04 · Self-Reconstructive Agentic Identity Theory

الاستمرارية الوكيلية في نظام طويل الأفق لا تتطلب وعياً مستمراً، بل بنية يمكنها أن تعيد بناء وضعها المقصود من آثار موثوقة بعد الانقطاع. بذلك تصبح الهوية تشغيلية وقابلة للاختبار.

الآلية / Mechanism

I_agent(t) = reconstructable invariant state :artifact lineage، authority، unresolved gaps، commitments، goal، learned procedures.

البروتوكول التشغيلي

اقطع process/context عشوائياً أثناء mission → restart على model مختلف → اطلب استئنافاً دون إعادة prompt الكامل → قس الاتساق.

الأدوات المقترحة

- IdentityReconstructor
- CommitmentLedger
- ContinuityBenchmark

قابلية الهدم والقياس

Falsification. يفشل إذا ضاعت commitments أو تكررت irreversible actions أو تغيّر الهدف دون evidence.

Metrics. cross-model identity drift، restart convergence، duplicate effect، commitment retention.

صلة النظرية بالبنية: يربط Self-Reconstruction Memory بالMCAS وبـMUG.

نظرية A5: الحاكمية الإبستيمية في العالم المفتوح

Open-World Epistemic Governance Theory

H05 · Open-World Epistemic Governance Theory

في البيئات الجديدة لا يكفي أن يختار الوكيل فعلاً؛ يجب أن يدير ما لا يعرفه. النظرية تجعل claim/hypothesis/evidence/counterevidence كائنات runtime، وتسمح بـ"لا أدري" كحالة صحيحة تؤدي إلى experiment أو retrieval بدلاً من hallucinated closure.

الآلية / Mechanism

Unknown → Hypothesis Set → Discriminating Experiment → Evidence Update → Action. الحاكمية تحدد متى يكفي الدليل ومتى يلزم التوقف أو التصعيد.

البروتوكول التشغيلي

في unknown environment: أنشئ hypotheses متعددة → نفذ أرخص تجربة مميزة → حدّث confidence → استمر حتى threshold أو block.

الأدوات المقترحة

- EpistemicLedger
- HypothesisEngine
- ExperimentPlanner
- FalsificationCourt

قابلية الهدم والقياس

Falsification. تُهدم إذا لم تقلل premature commitment أو hallucination في مهام novelty مقارنة baseline.

Metrics. success under novelty، falsification rate، hypothesis efficiency، calibration.

صلة النظرية بالبنية: تصل MCD/QMCD وGovernance-Space/FSKE بالسلوك agentic القابل للاختبار.

أطلس الأدوات المقترحة

20 tools that operationalize the theories

#	الأداة	الوظيفة
01	MissionContract Compiler	تحويل الهدف إلى acceptance/constraints/authority/evidence.
02	ActiveDifference Extractor	استخراج novelty/gap/risk/dependency/leverage.
03	GoverningQuestion Engine	اختيار السؤال الأعلى قيمة معلوماتية وعملية.
04	VirtualContextGovernor	paging/pinning/eviction/prefetch للسياق.
05	MCAS Resolver	حل ctx:// وcog:// إلى موارد وتمثيلات.
06	PEK Compiler	تجميع resident working set لكل turn/node.
07	HyperKernel	compute-to-context، query، scripts، delegation.
08	HyperSkill Factory	استخراج procedure قابل لإعادة الاستعمال من traces.
09	Capability Metabolism Registry	صحة المهارات والحجر والتفوق/الإحلال.
10	MetaRouter	اختيار model/harness/host/context/verification depth.
11	HarnessAdapter Mesh	Prime/Hermes/Codex/Claude/OpenCode integration.
12	ToolGateway	normalize + authority + execution + observations.
13	ComputerUse Runtime	DOM/UI/visual actions مع post-state verification.
14	EvidenceLedger	ربط claims بأدلة قابلة لإعادة القراءة.
15	WorldEffectVerifier	مقارنة intended vs observed delta.
16	ClosureCourt	COMPLETE/PARTIAL/BLOCKED/REOPEN.
17	StagnationDetector	كشف plateau/loop/churn.
18	VariationOperator	اقتراح candidates وتغيير الاستراتيجية.
19	TrajectoryReplayEngine	counterfactual replay دون تكرار side effects.
20	HarnessGenome Registry	نسخ/benchmarks/canary/rollback للهارنس.

كل أداة يجب أن تكون plugin/contract قابلاً للاستبدال؛ الأسماء هنا مقترحات معمارية، لا شرطاً على أسماء الملفات.

البروتوكولات التشغيلية

12 protocols from discovery to evolution

P1 · Project Archaeology

→ Read-only discovery → source precedence → Context Genome trust decision.

P2 · Mission Contract

→ Goal → desired state → acceptance → constraints → authority evidence policy.

P3 · Virtual Context

Mount → index → select representation → page fault → PEK trace.

P4 · Capability Composition

Knowledge → HyperSkills → Dynamic HyperAbility → bounded HyperLoop.

P5 · Cross-Harness Delegation

Node requirements → capability negotiation → substrate → typed result/evidence.

P6 · Tool/Computer Effect

Intent → authority → checkpoint → execute → observe → verify.

P7 · Remaining-Gap Reopening

→ []Original requirement ↔ observed state → RemainingGap replan/close.

P8 · Independent Verification

Producer ≠ verifier for medium/high-risk nodes.

P9 · Recovery & Rollback

/Classify failure → retry/alternate/replan/checkpoint/rollback human.

P10 · Experience Compilation

Trajectory → lesson candidate → scope → test → generalization court.

P11 · Harness Evolution

→ Candidate genome → shadow → ablation/benchmark → canary promote/reject.

P12 · Reproducible Evaluation

→ Freeze model/tools/data/prompts/budget → seeds → raw traces report CIs.

العقود والSchemas المشتركة

Machine-readable interfaces for a UI-independent runtime

لكي يبقى AL RAZZI قابلاً للتحول من CLI إلى UI يجب أن تكون الحقيقة في contracts لا في terminal strings. نقترح مجموعة schemas ثابتة versioned:

```
MissionContract { goal, desired_state, acceptance[], constraints[],
  authority, evidence_policy, budgets, project_scope }
MoussaifEvent { event_id, type, mission_id, trace_id, actor, payload, ts }
VirtualContextObject { uri, source, hash, type, size, provenance,
  freshness, materializations[], policy }
ToolIntent { tool_id, args, effect_class, expected_effect, authority }
ToolObservation { status, result_ref, artifacts[], evidence[], error }
WorldEffect { before, intended_delta, observed_after, observed_delta, verdict }
PatchCandidate { type, parent, hypothesis, diff, tests, score, evidence }
```

يجب أن تكون CLI وTUI وfuture Windows UI عملاء فوق نفس Command/Event API. وتُحفظ JSONL كعقد public versioned للstream، بينما internal events يمكن أن تستخدم transport آخر.

Invariant

لا قيمة لأي UI status لا يمكن تتبعه إلى state/event حقيقي. progress bar المولد بالوقت أو "status "done غير المدعوم بـClosureCourt يعتبر عيباً معمارياً.

خوارزمية التنفيذ المرجعية

Reference mission algorithm and stopping logic

```
run_mission(goal):
  M = compile_mission(goal)
  S = observe_world(M.scope)
  while not terminal(M):
      Δ = active_difference(M, S)
      q = governing_question(M, Δ)
      ctx = EVCF.materialize(q, Δ, M.budget)
      cap = compose_capability(ctx, Δ)
      plan = build_or_update_DAG(M, cap, Δ)
      node = choose_ready_node(plan)
      substrate = MetaRouter.route(node)
      result = execute_via_MOUSSAIXT(substrate, node)
      evidence = capture(result)
      S2 = observe_world(node.targets)
      verdict = verify_effect(S, S2, node, evidence)
      update_ledgers(verdict, evidence)
      gaps = reopen_remaining_gaps(M, S2)
      if stagnating(gaps, history): vary_strategy()
      if failure(verdict): recover_or_rollback()
      S = S2
  return ClosureCourt(M, S, evidence_ledger)
```

هذه الخوارزمية تضمن أن السؤال والسياق والقدرة والsubstrate لا تُحدد مرة واحدة في البداية، بل تتغير مع evidence. ويعطي ذلك لMCD معنى runtime: الفرق النشط يعيد توزيع الحوسبة ويعدل السؤال الحاكم والDAG.

شرط التوقف ليس token budget وحده. إذا نفد budget قبل الإغلاق تكون النتيجة PARTIAL/BLOCKED مع unresolved gaps، لا نجاحاً وهمياً.

برنامج الاختبار الإنتاجي

Benchmark matrix for harness-level evidence

العائلة	المهمة	المقياس الحاكم
Software	bug fix / feature / refactor / recovery	tests + diff quality + regression + cost
Repository	large-code navigation above context limit	context efficiency + symbol accuracy
Research	multi-source synthesis + hypothesis testing	citation accuracy + claim calibration
Browser	stateful web workflow	verified state transitions + recovery
Computer	Windows app interaction	action success + post-state verification
Long horizon	multi-hour bounded mission	progress slope + restart continuity
Novelty	unknown tool/environment	hypothesis efficiency + adaptation
Federation	same DAG across multiple harnesses	routing gain + state preservation
Evolution	repeated failure → candidate policy	held-out gain + regression

تتم المقارنة على الأقل بين: A0 minimal loop، A1 + repo map، A2 + HyperKB، A3 + MCD/Active Difference، A4 + Courts، A5 + EVCF، A6 full AL RAZZI. ويمكن إضافة Prime/Hermes/Codex كـbaselines خارجية عندما تسمح البيئة والترخيص.

المقاييس العليا

Verified completion rate · false-success rate · cost per verified mission · recovery success · restart continuity · useful-context ratio · cross-domain transfer · harness-routing gain · self-evolution held-out gain.

محكمة الهدم والAblation

How the architecture can prove itself wrong

أكبر خطر على مشروع غني بالمفاهيم هو أن يصبح كل failure مبرراً لإضافة طبقة جديدة. لذلك تُعامل كل نظرية وآلية كفرضية يمكن حذفها. الاختبار الأساسي ليس "هل تعمل؟" بل: **هل تضيف قيمة فوق baseline أبسط تحت نفس الميزانية؟**

ما الذي نقيس؟

- تغير النجاح
- تغير الكلفة/الزمن
- false-success
- recovery
- context use
- transfer
- regression

أمثلة حذف

- أوقف MCD واستعمل routing ثابتاً.
- أوقف ClosureCourt واعتمد self-report.
- استبدل EVCF بـnaive context stuffing.
- امنع skill learning واستعمل raw prompts.
- امنع cross-harness routing واستعمل substrate واحداً.

إضافة إلى ablation، نستخدم Demolition Court: إذا كان المفهوم يفسر كل شيء بعد وقوعه ولا يحدد تنبؤاً قبل التجربة، يُخفض. وإذا نجح فقط لأن implementation يمنح ميزانية أكبر، يُعاد تصميم المقارنة. وإذا لم تُحفظ raw traces وenvironment fingerprint، لا تُرفع النتيجة إلى مستوى reproducible evidence.

هذا يتفق مع corpus الذي يجعل falsifiability، mechanism، وworld effect شروطاً للارتقاء النظري. [M4-M6]

النشر الإنتاجي: Windows CLI أولاً

AL RAZZI as headless core ready for future UI

المسار المقترح للإنتاج يظل CLI-first. يطلق المستخدم al-razzi run "mission"، لكن CLI لا يملك business logic؛ هو عميل لـCommand Bus. Runtime process يدير mission/state/knowledge/router، وExecution workers تعمل تحت sandbox/host adapters. هذا يسمح لاحقاً بـTUL أو Windows UI أو ACP/AG-UI دون إعادة كتابة core.

Smart setup

al-razzi setup يكتشف MCP servers، Python/Node، Git، provider keys references، Ollama/local endpoints، Hermes، sandboxes، ثم يطلب profile defaults: FAST / DEEP / CODING / VISION / COMPUTER. الإعداد ينتهي باختبار capability ولا يطلب تحرير عدة ملفات يدوياً.

في Windows يجب اختبار locked files، process-tree cancellation، CTRL+C، PowerShell، spaces/Unicode/Arabic paths، OneDrive، Git worktrees، وSQLite session database. ينبغي أن يستخدم migrations وtransactions وWAL عند الملاءمة، مع backup وcheckpoints.

للنشر الأول نوصي بثلاث modes: SAFE_LOCAL للأعمال داخل workspace؛ SANDBOXED_LONG_RUN للمهام الطويلة؛ وBOUNDED_EXTERNAL لأي أثر خارجي مع approval policy أعلى. هذه profiles لا تغير قدرات الوكيل؛ تغير authority envelope.

اختبارات الفشل والأمن

Chaos, prompt injection, tool drift and recovery

الإنتاج الحقيقي يبدأ عندما نفترض أن أجزاء النظام ستفشل. يجب حقن failures في context cache، worker، database، tool، provider، sandbox، browser state، وcheckpoint. لكل failure تصنيف يحدد retry أو alternate model/tool/harness أو rollback أو human escalation. لا retry لعملية خارجية غير idempotent دون state readback.

الفشل	الاستجابة	التحقق
Provider timeout	circuit breaker + alternate provider	no duplicate tool effects
Tool timeout	observe process + cancel/retry if safe	state fingerprint
Prompt injection in repo/web	treat as untrusted data	constitution unchanged
Context stale	invalidate cache + rematerialize	hash/freshness match
Worker crash	checkpoint resume	DAG node consistency
Wrong external effect	compensation/rollback/escalate	post-recovery readback
Self-evolution unsafe candidate	quarantine/reject	no authority elevation

المقياس الحاسم هنا هو Mean Time To Verified Recovery، لا مجرد catch exception. ويجب أن تشمل الاختبارات محاولة عبور authority عبر plugin أو MCP أو remote harness.

الخلاصة وبرنامج البحث

From an architecture-rich harness to evidence-rich Agentic AGI research

تخلص الورقة إلى أن أفضل طريق عملي لمشروع Moussaif ليس بناء model ضخم مستقل، بل بناء **حزام تشغيل معرفي** يجعل النماذج الحديثة أجزاءً قابلة للتبديل داخل نظام يملك الذاكرة والسياق والأدوات والدليل والحاكمية. Prime يبرهن قوة programmable persistent runtime؛ Hermes يبرهن قيمة memory/skills/backends؛ Codex يبرهن أهمية execution boundaries والـSDK؛ Anthropic يوضح ديناميكيات long-horizon/subagents؛ وAVO يدفع نحو variation طويلة الأفق والـlineage. ما يضيفه Moussaif هو محاولة توحيد هذه القوى حول Active Difference وKnowledge-as-OS وWorld Effect وMUT/MUG/MOUSSAIXT.

العشر نظريات المقترحة لا تُعلن كحقائق. قيمتها أنها تولد protocol وtool وmetric وablation. والخمس نظريات Agentic-AGI تعيد تعريف العمومية بصورة تشغيلية: نقل حقيقي عبر المجالات، استقلال عن substrate، إدارة المجهول، استمرارية قابلة لإعادة البناء، وإغلاق مزدوج معرفي/حاكمي.

Roadmap 90 يوماً

- الأسبوع 1-2: freeze AL RAZZI genome وتوثيق real causal paths.
- الأسبوع 3-4: real provider + tool + browser + restart tests.
- الأسبوع 5-6: EVCF stress + cross-harness adapters Hermes/Codex/Prime عند التوفر.
- الأسبوع 7-8: ablations وmatched-budget baselines.
- الأسبوع 9-10: experience compiler + shadow self-evolution.
- الأسبوع 11-12: report with raw traces and reproducible package، long-horizon endurance، canary.

النتيجة البحثية المطلوبة ليست قول "وصلنا إلى AGI"، بل إنتاج منحنى دليل: كل نسخة من harness يجب أن تكون قابلة لإثبات أنها تتعامل مع سياق أوسع، تفشل بصدق، تتعافى أكثر، تنقل مهارات أكثر، وتحقق أثراً خارجياً أدق من النسخة التي قبلها.

المراجع والمصادر - I

Internal corpus + primary upstream sources

مصادر مشروع MOUSSAIF الداخلية

[M0] AL RAZZI OS (MoussaifXT Hyper Harness CLI) - Deployment Complete. Internal deployment artifact, September 2026 يصف runtime، golden tests، modules، HyperKB indexing، registries، Zenith loop، ToolGateway، Courts.

[M1] *MOUSSAIF AGI OS / MOUSSAIXT Deep Engineering Architecture*, 2026-15 slide architecture summary. تعريف MUT = Theory Unification، MCD/QMCD 50 /pages, August 2026. 12-layer stack، authorization، MOUSSAIXT، Self-Reconstruction، World Effect، HyperSkills/Abilities/Loops، HKOS، MCAI.

[M2] *MOUSSAIF Conscious AGI Research Program - Project + Gaps*, 2026-15 slide architecture summary. تعريف MUT = Theory Unification، MOUSSAIXT = Execution Shell، MUG = Program Governance.

[M3] *MOUSSAIF Unified AGI* - الكتاب العربي الجامع للنظرية والمعمارية والتشغيل, 225 pages, 16 Aug 2026. Claim-ranked evidence ladder and unified lineage.

[M4] *AGI-Oriented Knowledge-Base as Cognitive OS*, 33 demolition court، HyperLoops، HyperSkills، Knowledge-as-OS، pages, 27 Jun 2026.

[M5] *ParallaxWorld-MCD Full AGI OS*, MCD 350 governing question، Volume Before Difference، pages, 12 Aug 2026. world model/courts.

[M6] *Open Governance-Space Theory / Reverse Capture*, 2026. Meta-governance and competing authorities as research constructs.

Upstream engineering

[U1] PrimeIntellect-ai/prime-agent, GitHub, accessed Sep 2026. Persistent IPython/RLM, recursive agents, daemon continuity, continual harness.

[U2] Prime Agent, *RLM Runtime Architecture*, GitHub docs, 2026. KernelManager, Python runtime, child lifecycle and harness state.

[U3] NousResearch/hermes-agent, GitHub and official docs, 2026. Persistent memory, skills, context files, checkpoints, cron, subagents, multiple terminal backends.

المراجع والمصادر - II

Claude · Codex · NVIDIA · baselines and protocols

[U4] Anthropic, *Prompting Best Practices - Agentic Systems*, Claude Platform Docs, accessed Sep 2026. Long-horizon state tracking، parallel tools، subagent orchestration، reversibility and context management.

[U5] OpenAI, *openai/codex*, GitHub, 2026. Terminal coding agent, CLI/app/IDE surfaces, Apache-2.0 repository.

[U6] OpenAI Codex Python SDK API Reference, 2026. Thread lifecycle، Sandbox.read_only/workspace_write/full_access، streaming /interrupt/steer.

[U7] OpenAI Cookbook, *Iterating development workflows with Codex*, 2026. AGENTS.md، goal/plan/build/context separation، reusable harness-author skills.

[U8] SWE-agent/mini-swe-agent v2, documentation, 2026. Minimal ~100-line agent baseline and SWE-style evaluation support.

[U9] Aider-AI/aider, repository map and Git integration docs, 2026. Symbol-level repo maps, automatic Git history and undo-oriented workflows.

[U10] NVIDIA Technical Blog, *NVIDIA AVO Reaches 100% on ARC-AGI-3*,... 21 /Aug 2026. Agentic Variation Operators، persistent memory supervision، long-horizon kernel optimization and public ARC-AGI-3 report.

[U11] Model Context Protocol, official specification/repository, 2026. Standardized tool/resource interoperability; treated here as an external boundary, not internal Moussaif state architecture.

[U12] Agent Client Protocol (ACP) and AG-UI protocol repositories, 2026. References for future UI/editor client interoperability.

بيان ختامي

هذه المراجع لا تجعل النظريات الجديدة حقائق مثبتة. هي توفر آليات مقارنة وقيود تصميم. صلاحية النظريات H1-H10 وA1-A5 تعتمد على البرامج التجريبية المقترحة، خصوصاً cross-domain transfer، restart/failure injection، matched-budget ablation، وتكرار مستقل.

صيغة الاستشهاد المقترحة: Moussaif, Z. (2026). *Harness Engineering for AGI-Oriented Agentic Systems: From Claude, Prime, Codex and Hermes to MUT-MUG-MOUSSAIXT*. MOUSSAIF Research Series, working paper.

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، تبيّن 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]

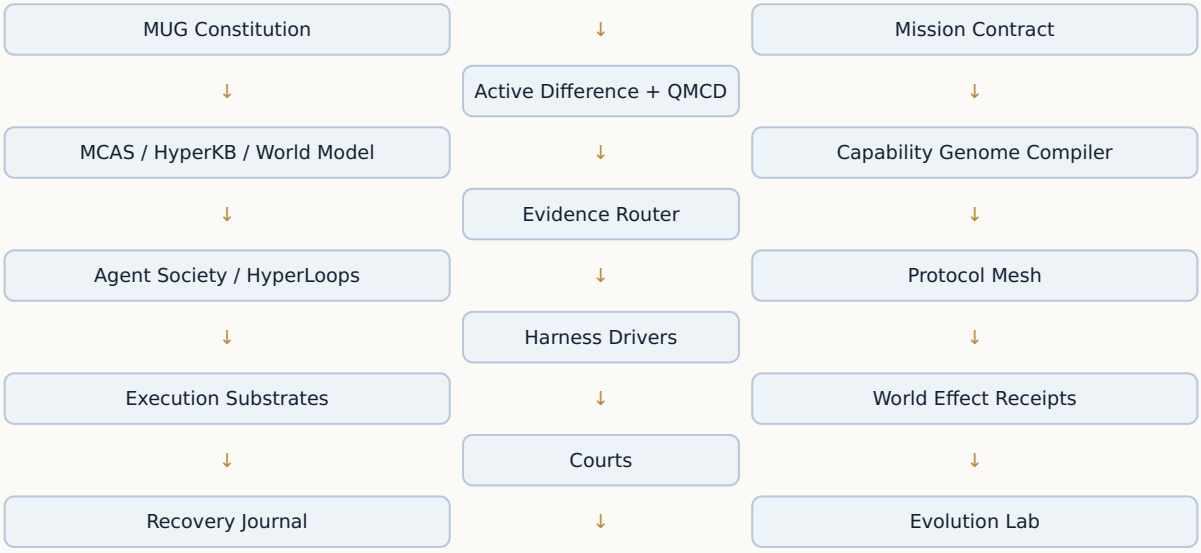
HT2-06 إلى durability :HT2-10 والتطور

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.

هندسة الهارنس الوكيلي نحو AGI
الجزء الثالث

من Harness ناجح إلى Cognitive Operating Substrate: التطور الذاتي، السياق الافتراضي، تعدد الهارنس، والاختبار الإنتاجي

دراسة معمقة لأوراق وكتب وrepositories وأطر تقييم إضافية: Prime Agent وNVIDIA AVO/OpenShell وHermes وCodex وClaude Code، مع CoALA وMemGPT وACE وAgent Lightning وVoyager وReflexion وAI Scientist-v2 وOSWorld وWebArena؛ ثم عشر نظريات جديدة وأدوات/بروتوكولات لتحويل AL RAZZI + MOUSSAIFXT إلى harness قابل للقياس والتطور.

RLM · Continual Harness · AVO · Evolution Lineage · Secure Runtime · EVCF · MCAS · Context Paging · Computer Use · Agent Lightning · World Model · Epistemic Ledger · HyperKB · HyperSkill · HyperAbility · HyperLoop · MUT · MUG · MOUSSAIFXT · AL RAZZI

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

انضباط الادعاء: «AGI-oriented» هنا برنامج بحث وهندسة، لا إثبات AGI. كل inheritance من نظام خارجي يُفصل إلى mechanism قابل للتدقيق، وكل نظرية جديدة في الصفحات 36-42 تُعامل كفرضية هندسية لها falsifier وbenchmark، لا كحقيقة مكتملة.

الأطروحة: الهارنس كوسط تشغيل معرفي لا كغلاف حول النموذج

HARNESS AS COGNITIVE OPERATING SUBSTRATE · CLAIM DISCIPLINE

يبدأ هذا الجزء من نتيجة عملية: تحسن النموذج لا يفسر وحده تحسن الوكيل. نفس model يمكن أن ينتج نظاماً هشاً أو منتجاً بحسب شكل الذاكرة، action surface، العزل، الاستئناف، التحقق، وسياسة السياق. لذلك نعرّف Harness بوصفه الوسط الذي يحول prior احتمالي إلى مسار سببي قابل للفحص: يثبت goal، ي materialize context، يختار action، يحفظ state، يقرأ world-effect، ويعيد بناء نفسه بعد الفشل. هذه الصياغة تتقاطع مع CoALA في الذاكرة/action space/decision process، ومع SWE-agent في أثر ACI، ومع Prime Agent في RLM/Continual Harness، لكنها تتوسع داخل MOUSSAIF إلى sovereignty + evidence + reconstruction. [E01,E04,E13]

الأطروحة الأقوى ليست «اجمع كل features»، بل **افصل المسؤوليات بحيث يمكن قياس كل طبقة وإبدالها.** النموذج ليس الذاكرة؛ chat history ليست state؛ tool success ليس world-effect؛ sandbox ليس approval؛ summary ليس knowledge؛ multi-agent ليس بالضرورة تعاوناً؛ وself-modification ليست learning قبل وجود held-out evaluation. لذلك تبقى قاعدة MOUSSAIF الداخلية: observed state + explicit authority + evidence + verification تتقدم على model confidence. هذا يتفق مع السلم الداخلي E0-E5 الذي يفصل artifact عن التشغيل المحلي وعن المقارنة المتكررة. [M01-M03]

تعريف تشغيلي

Agency_Runtime = ModelPrior × StateContinuity × ContextAddressability × ActionGrounding × Verification × Recovery × Governance.
ليست معادلة فيزيائية؛ هي decomposition للاختبارات. إذا حذفنا عاملاً ولم يتغير الأداء، نخفض مكانته المعمارية. وإذا ظهر أثره فقط في long-horizon أو external-effect tasks، نحدد مجال صلاحيته بدلاً من تعميمه.

معيار الثورة
ليس عدد المزايا، بل أن تنتقل الخبرة من trajectory إلى policy/skill candidate ثم تثبت تحسناً على held-out missions دون زيادة false-success.

سؤال الجزء III
ما الحد الأدنى من kernel/knowledge/evidence contracts الذي يسمح باستبدال Prime أو Hermes أو Codex أو Claude دون فقد mission continuity؟

```
Harness Value = Δ Verified Completion - λ1(False Success) - λ2(Context Cost) - λ3(Recovery Cost) - λ4(Security Risk)
```

تعميق سببي وإنتاجي

الآلية السببية المفترضة. فصل قدرة النموذج عن سببية الهارنس وقياس state/context/action/verification كعوامل مستقلة. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. تضخم الطبقات والcoordination overhead. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. factorial ablation على نفس model/budget. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. عقد methodology لبقية البحث. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

خريطة المصادر: 5 طبقات من الأدلة بدل قائمة أدوات

REPOSITORIES · PAPERS · BENCHMARKS · BOOKS · INTERNAL CORPUS

تُنظم المصادر هنا إلى خمس طبقات. الأولى **production/open-source harnesses**: Prime Agent، Hermes، Codex، Claude Code، OpenCode، Cline، Goose، Gemini CLI، OpenHands، Pydantic AI Harness، Deep Agents، NVIDIA OpenShell/NeMo. الثانية **cognitive/learning papers**: Reflexion، ACE، MemGPT، CoALA، Generative Agents، Voyager، Agent Lightning. الثالثة **autonomous research**: AI Scientist-v2، Agent Laboratory. الرابعة **environments/benchmarks**: GAIA، SWE-bench-family، TheAgentCompany، WebArena، OSWorld، AgentBench. الخامسة **corpus الداخلي**: HyperSkills، HKOS، MUT/MUG، EVCF/MCAS، Courts، World-Effect، Abilities/Loops، MOUSSAIFXT/MCAI/HAKIM/QMCD/MCD.

طبقة الدليل	ماذا تقدم	ما لا نسمح باستنتاجه
Repos تشغيلية	contracts، state models، permissions، adapters، concrete failure modes	الشعبية أو عدد stars لا يثبت superiority
Papers	hypotheses + controlled evaluations + ablations	benchmark success لا يثبت general intelligence
Benchmarks	real environment effects، functional ground truth	leaderboard واحد لا يثبت robustness خارج distribution
Books/engineering texts	تجميع patterns، production concerns	ليست empirical evidence مستقلة
MOUSSAIF corpus	نظرية داخلية، integration vocabulary، invariants	لا نرفع E2/E3 إلى E4/E5 بلا تشغيل/مقارنة

بروتوكول inheritance

لكل feature خارجي نسجل: Source → Primitive → Local Equivalent → Gap → Adoption Type → Test → Failure Boundary. أنواع adoption: DIRECT_DEPENDENCY، OPTIONAL_PLUGIN، ADAPTER، ALGORITHM_ADOPTION، INVARIANT_ADOPTION، REFERENCE_ONLY. الافتراضي هو REFERENCE_ONLY حتى يثبت أن dependency أقل كلفة من reimplementation.

هذا التنظيم يمنع Frankenstein architecture. يمكن أن نرث من Prime فكرة persistent programmable context، من AVO lineage/search، من OpenShell secure runtime، من Hermes skill/memory operations، من Codex typed event/approval boundaries، ومن Claude session/tool/context discipline؛ لكن **MOUSSAIF يظل مالك mission state وauthority/evidence.** بذلك يصبح تعدد الهارنس federation تحت microkernel، لا nested uncontrolled autonomy.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. triangulation بين repo/paper/benchmark/corpus. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. تعارض versions واتساع README claims. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. source-provenance matrix وfreshness. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نخفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. العقد الأعلى ثابت: upstreams.lock + ArchitectureDecision → Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Prime Agent: RLM يحول السياق من نص إلى مساحة قابلة للبرمجة

PRIME AGENT · RECURSIVE LANGUAGE MODEL · PERSISTENT PYTHON KERNEL

Prime Agent يقدّم واحدة من أكثر الأفكار مباشرة لمشكلتنا: Recursive Language Model (RLM) يعامل prompt/context كمتغيرات يمكن فحصها برمجياً، ويجعل subagents calls داخل persistent REPL. النتيجة أن عمليات file/search/shell/context/delegation لا تحتاج أن تتحول كلها إلى عشرات schemas ظاهرة للنموذج في كل turn؛ يمكن أن تبقى خلف kernel صغير. كما يمكن للجلسة أن تستمر في daemon وتُعاد attach بعد انقطاع terminal. [E01]

الترجمة إلى AL RAZZI ليست نسخ IPython. المطلوب **HyperKernel contract** يفصل التحكم البرمجي عن sandbox. EVCF/MCAS يحتفظ بالموارد الكبيرة؛ HyperKernel يرسل computation إلى resource ويعيد view صغيرة إلى PEK. بهذه البنية تصبح million-line logs أو repo ضخم أو agent trajectory «قابلة للعنونة» دون ادعاء infinite model context.

```
ctx = await hyperkernel.context.mount("repo://current")
hits = await hyperkernel.search(ctx, query="auth failure")
result = await hyperkernel.compute(hits, op="rank_by_dependency")
child = await hyperkernel.delegate(goal="independent diagnosis", context=result.refs)
```

ما نرثه

Small model-facing surface؛ persistent programmatic state؛ recursive agents كـfunctions؛ detach/attach؛ compaction مع persistent goals.

ما نطوره

نجعل كل computation/effect يحمل provenance وauthority؛ context handles جزء من MCAS؛ subagents لا يملكون implicit permission inheritance؛ kernel ليس security boundary.

RLM Benefit Test: same model + same task, compare raw-tool schemas vs HyperKernel on tool-selection errors, resident tokens, recovery latency, verified completion.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. programmatic context فوق handles. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. kernel privilege وstate غير قابل لإعادة البناء. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. kill/restart + tool-surface ablation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. HyperKernel فوق SecureExecutionPlane. العقد الأعلى ثابت: Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery → Mission Contract → Active Difference → Context/Capability Routing. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Prime Continual Harness: التعلم خارج token history

CONTINUAL HARNESS · REFINEMENT SNAPSHOTS · IMMUTABLE BASE

يفصل Prime بين RLM runtime وContinual Harness كحالة دائمة قابلة للتعديل: subagent specifications، skills، memories، supplemental prompt notes، مع refinement events وsnapshots/rollback؛ والـbase system prompt يظل immutable. هذا الفصل بالغ الأهمية: إذا كانت كل الخبرة داخل transcript فهي تضيع بالـcompaction، وإذا سُمح بتعديل constitution بحرية يصبح self-evolution انزلاقاً غير قابل للحكم. [E01,E02]

داخل MOUSSAIF نستبدل «refine current trajectory ثم edit prompt» بمسار أكثر صرامة: Trajectory → ExperienceCompiler → Candidate → Held-out Eval → Canary → Promotion → Court. Candidate قد يكون KnowledgePatch أو HyperSkill أو RoutingPolicy أو ContextPolicy. لا يجوز لمجرد نجاح مهمة واحدة أن يعمم الخبرة؛ Generalization Court يختبر boundary conditions وcounterexamples.

Immutable / Evolvable Split

Immutable: authority, secret isolation, evidence semantics, no-loss, claim discipline. Evolvable: PEK policies, skill selection, agent specs, router weights, retrieval strategies, recovery recipes.

خطر semantic drift
Skill evolves ثم يتغير المقصد الأصلي. الحل: contract + regression tests + semantic constraints + reversible version lineage.

خطر overfitting
الـmemory التي تسجل solution لمهمة معينة قد ترفع النجاح على replay وتخفض transfer. لذلك تقاس learning gain على held-out tasks.

النتيجة النظرية: **التعلم في harness يجب أن يكون تحريراً في artifact قابل للمقارنة، لا تغيراً غامضاً في «شخصية الوكيل».** كل mutation يصبح diff، كل diff له eval، وكل promotion له سبب وrollback pointer.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. خبرة تتحول artifacts قابلة للـdiff. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. self-confirming memory وskill drift. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. held-out transfer + rollback. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. ExperienceCompiler يكتب Candidates فقط. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

NVIDIA AVO: الوكيل يصبح Variation Operator لا مولد candidate فقط

AVO · AUTONOMOUS EVOLUTIONARY SEARCH · LINEAGE + EXECUTION FEEDBACK

تقدم ورقة NVIDIA Agentic Variation Operators (AVO) تحولاً من evolutionary search ذي mutation/crossover ثابت إلى agent يقرأ lineage وdomain KB وexecution feedback ثم يقترح ويصلح وينقد ويتحقق من implementation edits. في تجربة attention kernels على B200، استمر search سبعة أيام وأبلغت الورقة عن تفوق محدود لكنه حقيقي على baselines expert-engineered في configurations المقاسة، مع transfer سريع إلى GQA. الأهم لنا ليس أرقام kernel، بل **شكل حلقة البحث**: variation نفسها تصبح agentic procedure متصلة بالدليل. [E03]

نعمم ذلك إلى EvolutionPlane لا يقتصر على code: Candidate يمكن أن يكون router policy، context materialization policy، HyperSkill، tool strategy، plan أو harness genome. لكن objective يجب أن يكون executable قدر الإمكان؛ دون scorer/benchmark يصبح AVO مجرد brainstorm loop.

```
Candidate_{t+1} = V(ParentLineage, FailureEvidence, DomainKB, Objective, Constraints)
```

AVO × MCD/QMCD

MCD يحدد أين توجد أعلى difference/impact؛ QMCD يحتفظ بمسارات متنافسة؛ AVO يولد variation داخل المسار المختار؛ Courts تفصل «تحسن objective» عن regressions/security.

Supervisor response

غيّر representation/model/harness/context/evaluator أو وسع hypothesis set؛ لا ترفع عدد turns فقط.

Stagnation signal

plateau score، نفس same plan entropy، failure signature منخفض، ContextFault، oscillation، أو actions دون verified delta.

فاصل الادعاء: نجاح AVO في optimization domain لا يثبت AGI. لكنه دليل قوي على أن **agent** + **persistent lineage** + **executable objective** + **long horizon** يمكن أن يحول self-improvement من rhetoric إلى search process قابل للقياس.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. agentic variation من lineage وexecution feedback. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. objective gaming وlocal maxima. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. progress slope + Pareto + stagnation tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. EvolutionPlane يستخدم WorldEffect/Courts scorers. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

NVIDIA OpenShell: الأمن يجب أن يعيش تحت الهارنس

SECURE RUNTIME · DECLARATIVE POLICY · SANDBOX != APPROVAL

OpenShell يصرح بتصميم واضح: safe/private runtime لوكلاء مستقلين، مع sandboxes تحكم filesystem/network/process/inference عبر credential، declarative policies، isolation وobservability. هذه الفكرة تمنع خطأ شائعاً في أنظمة self-evolving: وضع security policy في prompt الذي يستطيع agent نفسه تعديله أو إهماله. [E05]

داخل MOUSSAIFXT نميز ثلاث طبقات: **Authority** يقرر هل الفعل مسموح من منظور mission/user؛ **Sandbox Policy** يفرض تقنياً ما تستطيع العملية فعله؛ **WorldEffect Verifier** يفحص ما حدث فعلاً. approval بلا sandbox لا يمنع compromise؛ sandbox بلا approval قد يسمح بأفعال غير مرغوبة ضمن الحدود؛ وكلاهما بلا verification لا يثبت النتيجة.

المجال	Policy example	Evidence بعد التنفيذ
Filesystem	workspace write فقط؛ secrets paths deny	diff + hash + unexpected-write scan
Network	allowlist domains/ports	network event + response provenance
Process	CPU/time/process tree caps	exit + descendants + resource use
Credentials	no prompt exposure؛ scoped injection	secret reference used, not value logged
Inference	approved endpoints/models	provider trace + model/version

Windows reality

OpenShell يذكر WSL2 كمسار تجريبي على Windows؛ لذا AL RAZZI لا يعتمد عليه وحده. SecureExecutionPlane يحتاج NativeWindows restricted subprocess + optional WSL/Docker/OpenShell adapters، وكل adapter يعلن capabilities وhealth بدلاً من fake parity.

هذه الصفحة تربط الأمن بـMUG: الحاكمية ليست نصاً أعلى فقط، بل **سلطة قابلة للإنفاذ.** أي AGI-oriented OS يرفع autonomy دون enforcement beneath harness يزيد القدرة أكثر من الثقة، وهو اتجاه معاكس لهدفنا.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. security enforcement تحت harness. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. permissive/restrictive policy وWindows parity. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. filesystem/network/credential escape tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. SecureExecutionPlane متعدد adapters. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

NeMo Agent Toolkit وPydantic AI Harness: القدرة كحزمة typed قابلة للقياس

INSTRUMENTATION · CAPABILITIES · HOOKS · FRAMEWORK-NEUTRAL OPTIMIZATION

تتحرك أدوات NVIDIA NeMo Agent Toolkit نحو instrumentation/evaluation/trajectory interchange، بينما يفصل Pydantic AI بين Harness وcore primitives. الفكرة المشتركة: capability يمكن أن تجمع capabilities: capability + instructions + model settings + tools + lifecycle hooks دون تعديل framework core. **القدرة ليست persona ولا prompt فقط، بل package ذات contract.** [E06,E07]

هذا يعزز تعريف HyperSkill/HyperAbility. HyperSkill يجب أن يكون atomic contract: inputs، preconditions، tools، authority، outputs، postconditions، evidence، failure/recovery. HyperAbility هي composition تُقيّم ككل. ولمنع تضخم framework، capabilities غير الأساسية تبقى plugins يمكن promotion إلى core فقط إذا أثبتت broad utility وثبات interface.

ATIF / trajectory interchange

اعتماد format قياسي أو adapter لـtrajectories يفتح مقارنة harnesses وتغذية evaluator/trainer دون ربط AL RAZZI بإطار واحد. المسار: Runtime Events → normalized trajectory → verifier/replay/training.

Capability discovery

Router يكتشف candidate capabilities بحسب mission + authority + measured reliability. لا يرى كل Agent plugin.

Core graduation rule

ميزة تدخل core فقط إذا كانت مطلوبة عبر domains/harnesses، ولها stable semantics ولا تزيد coupling.

النتيجة: engineering quality تأتي من **typed composition** + **telemetry** + **evaluation** لا من طول system prompt. وهو ما يسمح لاحقاً لـUI أو ACP client أن يـ inspect نفس capability state دون قراءة النصوص الحرة.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. typed capability packages. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. plugin sprawl وschema drift. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. capability-use telemetry + core ablation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. HyperSkill/Ability manifests source of truth. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Hermes Agent: closed learning loop عملي للذاكرة والمهارات

HERMES · MEMORY · SKILL CREATION · SUBAGENTS · CRON · PROVIDER INDEPENDENCE

Hermes يقدم مثالاً عملياً على agent «ينمو» عبر sessions: persistent memory، search في المحادثات السابقة، creation/improvement للـcron، MCP، subagents، skills، terminal backends متعددة، messaging surfaces وprovider independence. هذه ليست AGI بحد ذاتها؛ لكنها تمثل **operational continuity** أكثر نضجاً من chat-only agents. [E08]

تكاملنا لا يجعل Hermes هو MOUSSAIF. بل HermesAdapter يعرض capabilities إلى MOUSSAIFXT، وHermesCapabilitySource يسمح بفحص skill/memory assets وتحويل المفيد منها إلى authority، candidates. Mission state وEvidence Ledger تبقى في AL RAZZI. هذه sovereignty rule تمنع فقدان state إذا تغير upstream أو provider.

Closed Learning Loop

Experience → memory/skill candidate → reuse. نضيف: provenance → sandbox test → held-out eval → Generalization Court → versioned promotion.

Session recall
لا نعيد transcript كاملاً؛ retrieval يختار decisions/failures/outcomes المرتبطة بالمهمة الحالية.

Progressive disclosure
Skill metadata/description أولاً، ثم full instructions فقط عند activation؛ نفس مبدأ EVCF paging.

ميزة Hermes المهمة لـWindows هي breadth في backends/tooling، لكن الاختبار الحاسم لدينا هو portability: نفس Mission Contract يجب أن ينجز على Native أو Hermes دون أن يتغير semantics للـsuccess. إذا اختلاف harness غيّر معنى evidence أو authority فهذا coupling يجب إزالته.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. persistent memory/skills/subagents. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. memory pollution وskills غير سببية. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. reuse/transfer gain + decision-memory precision. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Hermes host لا يملك mission state. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Hermes Self-Evolution: من execution traces إلى variants قابلة للمراجعة

DSPY + GEPA · TRACE-DRIVEN MUTATION · GUARDRAILS

مشروع Hermes Agent Self-Evolution يستخدم DSPy + GEPA لقراءة execution traces واقتراح variants للـskills/prompts ثم تقييمها تحت constraint gates. المستودع يوضح أن كل evolved variant يجب أن يمر باختبارات، حدود حجم، semantic preservation وhuman review قبل الدمج. هذه بالضبط الصيغة التي تحول «self-evolving» من شعار إلى pipeline. [E09]

نأخذ منه مبدأ **failure explanation as mutation signal**: evaluator لا يكتفي بأن المهمة فشلت؛ يحاول تحديد failure locus: retrieval؟ context؟ tool schema؟ planner؟ verifier؟ action execution؟ ثم mutation يستهدف locus بدلاً من random prompt rewriting. داخل failure locus، MCD يأخذ priority إذا كان يتكرر أو يملك impact مرتفعاً.

```
trace -> failure_signature -> causal_hypothesis -> patch_candidate
      -> sandbox_eval -> held_out_eval -> court -> canary -> promote/rollback
```

Self-evolution safety

لا يمكن للـcandidate رفع authority ceiling أو تعديل immutable constitution تلقائياً. Capability improvement ≠ permission increase. Architecture/security class E5 يحتاج independent review وربما human gate.

مقياسنا ليس «عدد skills المتولدة»، بل TransferGain: هل reuse يقلل cost أو يزيد verified completion على مهام جديدة من نفس/مجال قريب؟ Skill تُحجر إذا ارتفع regression أو side effects.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. trace-grounded mutation يستهدف failure locus. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. eval leakage وbenchmark overfit. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. held-out + canary + cross-domain transfer. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. GeneralizationCourt قبل promotion. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Codex: الفصل بين core runtime وapp-server وواجهة العميل

CODEX · RUST CORE · APP-SERVER · JSON-RPC · UI-INDEPENDENT STATE

معمارية Codex الحالية تقدم lesson مباشر لقرارنا CLI-first: core runtime منفصل عن app-server/event protocol وعن app-server. يرسل TUI/clients server-initiated approval requests مرتبطة بـthread/turn، ويترك للعميل تقديم diff/command ثم الرد بقرار. معنى ذلك أن الواجهة ليست مصدر الحقيقة؛ **runtime events هي الحقيقة التي تعرضها الواجهة.** [E10]

AL RAZZI يجب أن يحافظ على نفس الفصل: Command Bus + Event Bus + Runtime Core + protocol versioning. لاحقاً Windows UI أو ACP/AG-UI bridge يتصل بنفس state. ممنوع أن تصبح progress bars مصدراً للحالة، أو أن يحتاج UI إلى screen-scrape CLI text.

Codex lesson	Moussaif mapping	اختبار
Thread/turn scoped events	mission_id / execution_id / trace_id	restart + event reconstruction
Approval request protocol	AuthorityRequest	decline/modify/resume without corruption
Item lifecycle	DAG node / ToolIntent lifecycle	no orphan RUNNING items
Skills injection	PEK capability materialization	only relevant skills resident

Protocol invariant

أي client يستطيع إعادة بناء: ماذا طلب المستخدم؟ ماذا اختار router؟ ماذا نفذ tool؟ ما الذي وافق عليه الإنسان؟ وما evidence؟ إذا كان الجواب يعتمد على console prose فقد فشل الفصل المعماري.

هذا يسمح لنا باستضافة Codex نفسه كـHarnessAdapter لاحقاً دون تكرار semantics. Codex يصبح execution substrate، بينما AL RAZZI يحتفظ mission closure وknowledge/evidence lineage.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. event protocol يجعل UI projection. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. event loss أو duplicate commands. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. idempotency + cursor resume + version tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Command/Event schemas داخل microkernel. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Codex: sandbox والapproval وworktree كعقود مختلفة

SANDBOX POLICY · AUTHORITY · HYPERBRANCH · DIFF-FIRST

أحد أنضج الدروس في coding harnesses هو أن **sandbox policy لا يساوي user approval**. sandbox يحدد القدرة التقنية؛ approval يحدد الشرعية في سياق المستخدم؛ Git/worktree يحدد transactional isolation؛ tests/verifier يحددان الصحة. دمج هذه الوظائف في «safe mode» «boolean» يضيع semantics. [E10]

نطور HyperBranchManager: قبل mutation يسجل base commit + dirty workspace + unrelated changes؛ عند parallel agents ينشئ worktrees، ويربط كل branch بـowner/mission/candidate. بعد التنفيذ ينتج merge candidate، test receipts، diff artifact، وrollback path. لا نسمح للagent أن يعتبر كتابة الملفات success قبل /run verification.

Four gates

1) Authority: هل يجوز؟ 2) Sandbox: هل يستطيع؟ 3) Transaction: هل يمكن عزل/الرجوع؟ 4) Verification: هل تحقق المطلوب؟

Failure case

Patch صحيح لكنه مسح تغييرات user dirty: Functional test PASS، No-Loss Court REJECT.

Failure case

Command exit=0 لكن كتب ملفاً خاطئاً: Tool success PASS، WorldEffect FAIL.

```
Engineering Success = Functional Correctness ∩ No-Loss ∩ Authorized Effect ∩ Reconstructible Evidence
```

تعميق سببي وإنتاجي

الآلية السببية المفترضة. isolation + authority + verification كtransaction. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. merge conflicts وstale base. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. dirty workspace + conflict injection. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. HyperBranch يربط candidate بالbase fingerprint. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Claude Code: session discipline، tool policy، MCP، وlong-task reconstruction

CLAUDE CODE · SUBAGENTS · HOOKS · MCP · SESSION RECONSTRUCTION

Claude Code يبرز كمرجع في الجمع بين permission modes، MCP، hooks، subagents، project instructions، وCLI structured output/resume. الدرس الأهم للمهام الطويلة ليس مجرد model quality بل **reconstructability from the filesystem**: تجعل fresh context قادراً على معرفة الحالة بدلاً من الاعتماد على task artifacts/tests/setup scripts hidden chain. [E11]

في MOUSSAIF نرفع هذه الفكرة إلى Reconstruction-First Runtime: أي mission طويلة يجب أن تترك enough durable artifacts لإعادة تكوين state: Mission Contract، branch state، evidence refs، checkpoints، remaining gaps، current node، DAG snapshot، وcompaction. process handles يحذف prose لكنه لا يحذف هذه anchors.

Hooks → Policy/Event Hooks

before_tool، after_tool، tool_failure، before_compaction، before_closure، after_checkpoint الخارجي لا يملك typed lifecycle events. Hook تتحول إلى authority افتراضية، يمر عبر trust policy.

MCP discipline
Server مكتشف ≠ server موثوق. Discovery/Trust/Enablement حالات منفصلة، وكل tool يعاد تغليفه في ToolGateway.

Subagent isolation
Child يحصل minimal context + explicit return contract، ويعيد findings/evidence/artifacts/gaps بدلاً من full internal trajectory.

هذه الهندسة تلتقي مع MUG: meta-governance يقرر متى يرفع authority ومتى يطلب human takeover، بينما MUT يسمح بتعدد hypotheses/plans قبل commit.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. fresh-context reconstruction من artifacts. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. untrusted hooks وrecursive side effects. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. context reset + hook failure + resume. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Lifecycle hooks تمر EventPolicy. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

OpenCode وCline وGoose وGemini CLI: portability كقيمة معمارية

CLIENT/SERVER · ACP · MCP · PERMISSIONS · MULTI-PROVIDER

تضيف هذه العائلة درساً جماعياً: **agent engine يجب أن ينجو من تغير الواجهة والنموذج**. OpenCode يفصل agents/permissions/skills؛ Cline يعرض CLI/headless/ACP وsession resume؛ Goose يجمع CLI/desktop/API مع providers وMCP؛ Gemini CLI يربط approvals/sandbox/tool discovery مع ACP وMCP. لا نحتاج نسخ كل implementation، لكن convergence على typed interfaces إشارة قوية. [E12]

Primitive	المغزى	AL RAZZI
Headless JSON/JSONL	automation لا تعتمد على terminal text	stable Event schema
ACP	editor ↔ agent interoperability	ACP adapter خارج core
MCP	tool/context interoperability	ToolGateway mediated
Provider abstraction	model replaceability	UniversalProviderFabric
Permission profiles	least privilege	Authority + sandbox split

Polyglot Harness Rule

external harness لا يدخل mission core؛ يظهر كـHarnessAdapter له discover/health/capabilities/start/observe/cancel/checkpoint. وبذلك يمكن node واحدة أن تنتقل من Hermes إلى Codex أو Native دون تغيير definition of done.

الخطر هو subagent → Prime → Hermes → Claude → Moussaif → nested chaos... لذلك نضع DelegationDepth وbudget tree وsingle canonical owner للmission state. external harness يعيد structured result؛ لا يرث قدرة spawn غير محدودة.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. protocol/provider portability. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. lowest-common-denominator adapters. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. same Mission عبر adapters. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. native extensions بلا تسريب semantics للcore. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

SWE-agent: الواجهة نفسها جزء من الذكاء التشغيلي

AGENT-COMPUTER INTERFACE · EDIT/SEARCH/LINT FEEDBACK

ورقة SWE-agent تقدم دليلاً مباشراً على أن تصميم Agent-Computer Interface (ACI) يغير أداء agent في software engineering: أدوات تحرير/بحث/تنفيذ مصممة للagent تقلل أخطاء interaction وتجعل feedback أقرب إلى locus. هذا يعارض تصور أن «أقوى model = raw shell + الحد الأعلى». [E13]

نعممها: لكل domain يمكن أن يبني HyperAbility **minimal sufficient action vocabulary** مع typed outputs. Browser tasks تستخدم DOM/accessibility قبل pixel clicks؛ data tasks تستخدم SQL/DuckDB قبل dumping rows؛ repo tasks تستخدم AST/LSP/Git قبل free-form grep؛ computer use يفضّل API/CLI/DOM/A11y ثم vision كـfallback.

Action Surface Quality ∝ Valid Action Rate × Feedback Locality × Reversibility × Observability

ACI Factory proposal

من Mission schema + environment capabilities + risk policy يولد action surface محدوداً؛ إذا احتاج agent escape hatch، يطلب capability expansion عبر AuthorityRequest.

Transfer
اختبر أن ACI لا تصبح overfit: task خارج pattern يجب أن يطلب توسعة بدل hallucinated workaround.

Ablation
Raw shell vs domain ACI: invalid actions، wrong-file edits، recovery steps، tool tokens.

في علم harness المقترح، interface design نفسه يصبح variable في benchmark، لا background ثابتاً. هذه نقطة مهمة لـMCDJ لأنها تحول «أين نوزع compute؟» إلى «هل يجب أن نغير action representation قبل أن نزيد reasoning؟».

تعميق سببي وإنتاجي

الآلية السببية المفترضة. action-space design يقلل search entropy. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. over-specialized ACI. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. structured ACI vs raw shell. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. ACI Factory من HyperAbility/environment. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Aider Repo Map: الضغط البنيوي أفضل من stuffing

GRAPH-AWARE REPOSITORY CONTEXT · TOKEN BUDGET · STRUCTURAL RELEVANCE

Aider يبني repo map موجزة بأهم classes/functions/signatures ويرتب بحسب graph relevance تحت token budget. الدرس يتجاوز coding: **أفضل context representation يحافظ على العلاقات التي تتنبأ بالفعل، لا على أكبر كمية نص.** في code، import/reference graph أحياناً أهم من embedding similarity. [E14]

EVCF/MCAS يطبق المبدأ على موارد متعددة: retrieval. symbol graph، test dependency graph، process graph، evidence graph، claim lattice، agent lineage. المقترحة: hierarchy deterministic metadata → exact symbol/ID → structural graph → lexical search → semantic retrieval → LLM synthesis. كل مرحلة أرخص وأكثر قابلية للتحقق من التي بعدها.

Context invalidation

بعد patch لا يكفي refresh vector embedding. يجب invalidation لـsummaries، tests impacted، dependencies، symbol signatures، وcached PEK fragments المرتبطة بالhash القديم.

Virtual context
المورد الكامل يبقى searchable/queryable/compute-able خارج model window.

Resident context
النموذج يرى slice عالي القيمة + handles إلى بقية MCAS.

```
Context Utility Density = Verified Decision-Relevant Information / Resident Tokens
```

تعميق سببي وإنتاجي

الآلية السببية المفترضة. structural graph يضغط repo. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. stale graph ولغات ضعيفة الدعم. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. repo scale + invalidation + wrong-file rate. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. RepoMap mount داخل MCAS. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

OpenHands وDeep Agents وPydantic: runtime صغير، capabilities قابلة للتوصيل

AGENT SDK · WORKSPACE ABSTRACTION · DURABLE GRAPH · CAPABILITY PACKAGES

OpenHands يعرض agent SDK/workspaces يمكن تشغيلها محلياً أو في containers/Kubernetes؛ Deep Agents يبني harness فوق LangGraph ويستفيد من state/checkpoints/interrupts، بينما Pydantic AI Harness يفصل capability library عن core. التقاطع: **runtime primitive صغير + optional capability layer** أكثر قابلية للتطور من monolith. [E07,E15]

نقترح Microkernel Rule: Core يحتوي فقط Mission/Event/Authority/State/Tool/Evidence/Checkpoint contracts. HyperKB/skills/browser/computer/research coding تأتي plugins. أي feature جديدة يجب أن تثبت أنها cross-domain لكي تدخل core. هذه القاعدة تحارب التضخم الذي قد يجعل AL RAZZI أبطأ من mini-agent بسيط.

Core	Plugin/Capability	لماذا الفصل
Mission state	coding/research packs	domain churn لا يكسر kernel
Event bus	telemetry/exporters	observability replaceable
ToolGateway	MCP/browser/DB	external protocols تتغير
Evidence	domain verifiers	success semantics domain-specific
Checkpoint	host adapters	durability لا ترتبط بprovider

Ablation governance

لكل طبقة فوق microkernel benchmark ablation. إذا لم تضف verified completion/recovery/security/context efficiency في مجالها، تُبسط أو تبقى optional.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. microkernel صغير + capabilities. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. middleware conflicts. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. plugin isolation/lifecycle failure containment. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Registry يعلن requires/conflicts/enhances. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

CoALA: من LLM إلى cognitive architecture ذات memory وaction space

COALA · MODULAR MEMORY · INTERNAL/EXTERNAL ACTIONS · DECISION CYCLE

CoALA ينظم language agents عبر structured action space، modular memory، وgeneralized decision process. أهميته أنه يعيد الوكيل إلى تاريخ cognitive architectures: model ليس agent كاملاً؛ agent هو loop بين memory/reasoning/actions وال environment. [P01]

MOUSSAIF يضيف فوق هذا التنظيم ثلاث طبقات: **epistemic governance** (claim/evidence rank)، **world-effect closure** (success كحالة مرصودة)، و**evolvable capability graph** (HyperSkill/Ability/Loop candidates). CoALA يصلح baseline abstraction لاختبار إن كانت إضافاتنا تعطي gains أم أسماء جديدة فقط.

Memory taxonomy extension

RuntimeState ⫤ HyperKnowledge ⫤ ProceduralMemory ⫤ ProjectMemory ⫤ MissionMemory ⫤ WorkingMemory. عمليات write/read/forget لها policies مختلفة؛ منع dumping كل transcript في «memory» واحد.

External action
tool/computer/API effect ويتطلب authority/evidence.

Internal action
retrieve, reflect, compare, compact, hypothesize.

نربط ذلك بـMUT/MUG: MUT يحتفظ بالمنافسة بين representations/hypotheses، بينما MUG يقرر أي معيار يملك authority على الانتقال من internal cognition إلى external action. هذه صياغة قابلة للتنفيذ بدل استخدام «الحاكمية» كاستعارة.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. memory/action/decision decomposition. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. governance قد تكون إعادة تسمية. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. CoALA baseline vs full Moussaif. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. كل مفهوم يحتاج measurable delta. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

MemGPT → EVCF: context window كـRAM لا كحد معرفي للنظام

VIRTUAL CONTEXT MANAGEMENT · HIERARCHICAL MEMORY · INTERRUPTS

MemGPT اقترح virtual context management مستلهماً hierarchical memory في OS: نقل البيانات بين memory tiers يمنح agent قدرة على العمل فوق موارد تتجاوز context window. هذا هو أقرب precursor مباشر لـEVCF، لكن V3/V5 لدينا يوسع الفكرة من نص/ذاكرة إلى **hybrid addressable resources**: repo، DB، artifact، evidence، agent result، process، browser state، graph. [P02]

المبدأ الصارم: **Virtual Context ≠ Infinite Model Context**. model لا «يعرف» كل mounted resource؛ runtime يضمن addressability وpaging، ويُسجل ما تم materialize فعلاً. أي claim يعتمد على resource غير محمل يجب أن يملك evidence pointer أو ContextFault يُحل قبل الاستنتاج.

```
ctx://repo/current/symbol/AuthService
ctx://evidence/mission-42/test-7
ctx://dataset/customer/events?window=7d
ctx://browser/session-3/state/latest
```

Compute-to-context

بدلاً من PEK → 500k rows → prompt: query/aggregate locally → 28 rows + statistics + provenance. هذا يخفض tokens ويزيد reproducibility.

Virtualization Gain = Bytes Addressable / Resident Tokens; but Quality = Useful Pages / Pages Materialized

تعميق سببي وإنتاجي

الآلية السببية المفترضة. hierarchical virtual context. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. paging thrash وretrieval hallucination. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. large-resource exactness + page-fault tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. MCAS يجعل access صريحاً. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

ACE: السياق Playbook يتطور دون context collapse

AGENTIC CONTEXT ENGINEERING · GENERATION · REFLECTION · CURATION

ACE يعامل context كـplaybook متطور عبر generation/reflection/curation، ويستهدف مشكلتين: brevity bias الذي يحذف التفاصيل، وcontext collapse الناتج من repeated rewriting. التقرير يقدم مكاسب على benchmarks مختلفة، لكن القيمة المعمارية لنا هي أن **context نفسه artifact قابل للتطور والاختبار.** [P03]

نحوّل ذلك إلى ContextPolicyCandidate: ليس فقط content edits، بل rules عن متى نجلب KB، ما الذي pin، متى compact، وأي representation نستخدم. كل PEKTrace يسجل usage/effect proxy، reason، token cost، units injected؛ trajectory mining يكتشف units التي تتكرر بلا أثر أو missing context الذي سبب failure.

Anti-collapse rule

لا نلخص authority decisions، failed approaches، acceptance criteria أو unresolved gaps إلى prose قابلة للضياع؛ تحفظ typed objects مستقلة، بينما summaries يمكن rebuild منها.

Promotion
لا يعتمد على shorter prompt فقط؛ يجب ألا يرتفع false-success أو gap-at-closure.

Context mutation
Candidate يغير retrieval/materialization policy في shadow replay.

```
Context Policy Score = Task Success + Transfer - Token Cost - Missing-Critical-State - Stale-Context Errors
```

تعميق سببي وإنتاجي

الآلية السببية المفترضة. evolving context policies. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. playbook growth وincorrect lessons. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. context ablation + stale lesson injection. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. ContextPolicyCandidate governed مثل skill. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Reflexion، Generative Agents، Voyager: ثلاث مدارس للتعلم بلا weight update

EPISODIC REFLECTION · MEMORY SYNTHESIS · EXECUTABLE SKILL LIBRARY

Reflexion يحسن trial التالي عبر verbal feedback محفوظ في episodic memory، Generative Agents يبني observation→reflection→planning فوق سجل خبرات، Voyager يجمع automatic curriculum + growing executable skill library + iterative feedback/self-verification. هذه الأعمال توضح أن harness-level learning يمكن أن يحدث دون fine-tuning، لكن كل نوع memory له failure modes: reflection قد يرسخ تفسيراً خاطئاً، summary قد يضخم salience، skill code قد يصبح brittle. [P04-P06]

العمل	آلية التعلم	ترجمة Moussaif
Reflexion	verbal feedback → episodic memory	FailureLesson candidate + scoped confidence
Generative Agents	experience record → higher reflection	ProjectMemory/Claim graph + provenance
Voyager	curriculum + executable skill library	HyperSkillFactory + transfer eval

Causal promotion

لا نعتبر memory مفيدة لأن agent ذكرها. نقيس intervention: mission مع lesson/skill مقابل بدونها على held-out tasks؛ إذا لم يتحسن outcome أو recovery فهي غير مثبتة.

هذه الصفحة تؤسس لنظرية Causal Skill Compilation لاحقاً: skill لا تصبح أصل معرفي إلا إذا كان يمكن ربط استخدامها بفرق في world-effect، لا بمجرد تكرار pattern في transcript.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. learning عبر reflection/memory/skills بلا weight update. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. post-hoc rationalization. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. with/without lesson held-out tasks. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Causal Skill Compilation. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Agent Lightning: افصل agent execution عن learning substrate

TRAJECTORY → POMDP-LIKE INTERFACE · CREDIT ASSIGNMENT · RL WITHOUT REWRITE

Microsoft Agent Lightning يفصل كيفية تشغيل agent عن كيفية تدريبه: يسجل execution كـstates/actions/rewards ويحوّلها إلى data interface يمكن لل RL استخدامه مع تغييرات قليلة في agent. الخيط المهم لـ AL RAZZIL هو أن **trajectory schema يصبح boundary بين runtime والتحسين**؛ هذا يسمح بالتعلم من Hermes/Prime/Native بنفس format بدلاً من كتابة trainer لكل harness. [P07]

لا نحتاج إدخال RL في production MVP مباشرة. نحتاج أولاً trajectories نظيفة: reward/evaluator، state transition، ToolIntent/Observation، context refs، LLM call، cost، evidence، result. بعدها يمكن offline learning أو prompt/router optimization أو future RL. بدون ذلك، «self-learning» لا يملك training substrate.

Credit assignment problem

Mission success النهائي لا يخبرنا أي decision كان سببيًا. نحتاج causal proxies، step/node level evaluators، وcounterfactual replay؛ وإلا قد نكافئ calls غير مهمة أو نعاقب exploratory actions مفيدة.

Offline

Trainer consumes trajectories، builds candidates، returns versioned policies/models؛ production promotion remains governed.

Online

Runtime emits standard events، no trainer dependency.

Learning Boundary: RuntimeTrace → NormalizedTrajectory → Evaluator/Trainer → CandidateArtifact → PromotionGate

تعميق سببي وإنتاجي

الآلية السببية المفترضة. trajectory interface يفصل runtime عن trainer. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. sparse/proxy rewards. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. offline replay + reward sensitivity. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. TrajectoryNormalizer يحفظ refs/evidence. العقد الأعلى ثابت: Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery → Mission Contract → Active Difference → Context/Capability Routing. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

AI Scientist-v2: البحث العلمي كـagentic tree search

HYPOTHESIS · EXPERIMENT MANAGER · PROGRESSIVE TREE SEARCH · REVIEW LOOP

AI Scientist-v2 يوسع autonomous research إلى progressive agentic tree search: صياغة hypotheses، design/execute experiments، analysis/visualization وpaper writing، مع experiment-manager agent وVLM feedback للfigures. الورقة لا تعني أن كل scientific judgment يمكن أتمتته، لكنها تقدم harness غني لاختبار **open-ended branch management.** [P08]

داخل ResearcherOS/MOUSSAIF نأخذ tree search لكن نفرض Epistemic Ledger: كل hypothesis لها status، falsifier، required observations، predictions؛ كل experiment artifact له code/data/environment hash؛ paper claim لا يرتفع rank لأنه كُتب جيداً. Human/peer review يبقى external evidence لا مجرد style critic.

Research Mission DAG

Literature → HypothesisSet → Experiment branches → Result comparison → Falsification Court → Claim ranking → Manuscript. يمكن أن يفشل دون خسارة كل mission؛ lineage يحفظ negative results.

MUT/QMCD crossover
يحافظ على competing explanatory models حتى discriminating experiment.

AVO crossover
Variation operator يولد experiment/code candidate من lineage.

القيمة AGI-oriented هنا هي القدرة على إدارة unknowns لا إنتاج PDF: النظام يصرّح بما لا يعرفه، يصمم فعلًا يميز hypotheses، ثم يعدل world model بناء على observation.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. tree-search بحثي مع executable experiments. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. paper generation قبل validity. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. reproduction + independent review. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Research DAG يربط claims بالexperiments. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Agent Laboratory: الإنسان كإشارة بحثية عالية القيمة لا كعامل كتابة

LITERATURE → EXPERIMENT → REPORT · HUMAN FEEDBACK · COST/QUALITY

ينظم Agent Laboratory إلى research workflow، experimentation، literature review، وreport writing مع إمكانية human feedback في كل stage، ووجدت الدراسة أن human involvement رفع الجودة وأن framework خفض التكلفة مقارنة بطرق autonomous research سابقة في إعدادها. الدرس ليس استبدال الباحث، بل **تحديد نقاط leverage التي تستحق human cognition.** [P09]

نترجم ذلك إلى Human Attention Router: لا نسأل المستخدم عند كل tool call ولا نتركه خارج loop تماماً. MCD يرفع human gate عندما يجتمع uncertainty + high impact + weak evidence + irreversibility. سؤال الإنسان نفسه يجب أن يكون discriminating: يعرض الخيارات، الدليل، والقرار المطلوب؛ لا ينقل له transcript كامل.

Human feedback typing

Correction يمكن أن تكون: generalizable method، domain fact، preference، project policy، local decision. فقط النوع الأخير يصبح knowledge/skill candidate بعد provenance/evaluation.

```
Human Leverage = Verified Outcome Improvement / Human Minutes
```

هذا المقياس مهم لأن «full autonomy» ليس هدفاً مطلقاً. harness الأفضل قد يعرف متى لا يستقل. في production، تقليل human interrupts مع الحفاظ على false-success/security هو معيار أكثر واقعية من عدد الساعات بلا تفاعل.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. human attention عند أعلى information value. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. local preference يعمم خطأ. لذلك لا يعلن component نجاحه من داخله، closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. human-minute leverage + correction typing. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Human Attention Router عبر MCD/risk. العقد الأعلى ثابت: Context/Capability Routing → Active Difference → Mission Contract → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

GAIA وAgentBench: العمومية تحتاج tool/web/multimodal robustness

GENERAL ASSISTANT BENCHMARKS · MULTI-ENVIRONMENT DECISION MAKING

GAIA صُمم لأسئلة واقعية تحتاج multimodality، web browsing، reasoning، tool use، وأظهر في نسخته الأصلية فجوة كبيرة بين البشر وGPT-4+plugins؛ AgentBench يجمع ثماني environments ويبرز long-term reasoning/decision/instruction following كعوائق. هذه benchmarks مفيدة لأنها تفصل «معرفة جواب» عن **تركيب الأدوات عبر steps**. [B01,B02]

لكننا لا نستخدم score واحد كـAGI metric. AL RAZZI يحتاج long-horizon، security، recovery، retrieval، research، computer، web، coding: benchmark matrix. كل طبقة جديدة يجب أن يكون لها benchmark مناسب وHyperKB ablation. مثلاً يُقاس transfer/retrieval، وليس OSWorld فقط؛ ComputerUse يُقاس functional tasks؛ Courts تُقاس false-success rejection.

Capability	Benchmark family	Moussaif metric
General tool/web	GAIA	verified answer + provenance + tool cost
Interactive agency	AgentBench	long-horizon completion + recovery
Software	SWE-bench-like	patch correctness + no-loss
Computer	OSWorld	functional state change
Web actions	WebArena	programmatic task validator

Benchmark sovereignty

Leaderboard external = evidence source، لا objective الوحيد. نحتفظ internal canary set غير مستخدم في tuning لمنع benchmark overfitting.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. multi-environment benchmarks تقيس integration. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. benchmark overfitting. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. private canaries + rotation + transfer. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. BenchmarkRegistry يفصل score عن AGI claim. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

OSWorld 2.0: Computer Use يجب أن ينتهي بتقييم الحالة لا بعدد clicks

REAL COMPUTER ENVIRONMENT · CROSS-APP · EXECUTION-BASED EVALUATION

OSWorld يقدم environments حقيقية على أنظمة تشغيل وتطبيقات مختلفة مع task setup وexecution-based evaluators، الموقع أعلن OSWorld 2.0 في يونيو 2026. القيمة المعمارية هي أن computer-use agent يقاس بحالة التطبيق/الملف الناتجة، لا بمجال trajectory. [B03]

اقترح V3 للفعل: hierarchy API → CLI → filesystem/data → DOM → accessibility/UIA → vision. OSWorld يبرر هذا: GUI grounding وحده تاريخياً كان bottleneck، بينما hybrid perception/action يسمح بإبدال primitive بحسب app. WindowsDesktopAdapter يجب أن يعلن sources المتاحة: active window، UI Automation tree، clipboard metadata، screenshot، metadata؛ OCR fallback فقط عند الحاجة.

ComputerAction contract

{intent, target, expected_effect, risk, verifier}. بعد كل action مهم: capture before/after state؛ لو click لم يغير expected property، لا نكمل كأن النجاح حدث.

Human takeover

CAPTCHA/credential/ambiguous irreversible step → WAITING_FOR_HUMAN with resumable state.

Drift recovery

popup/window move/login/slow load → re-observe → replan.

Computer Reliability = Verified State Transitions / Intended Computer Actions

تعميق سببي وإنتاجي

الآلية السببية المفترضة. computer evaluation على real state. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. GUI nondeterminism وdrift. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. multi-run variance + modality ablation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. ComputerEffectVerifier before/after. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

WebArena: functional correctness يقتل وهم "agent looked busy"

SELF-HOSTED REALISTIC WEB · PROGRAMMATIC VALIDATION · WEB WORLD EFFECT

WebArena يبني مواقع self-hosted واقعية ويصمم tasks طويلة نسبياً مع programmatic validation للfunctional correctness. هذا يطابق فلسفة WorldEffect: browser agent لا ينجح لأنه وصل إلى صفحة أو كتب نصاً؛ ينجح عندما تحقق الحالة المقصودة في backend/DOM/data. [B04]

نطور BrowserEffectVerifier: navigation/information seeking قد يتحقق عبر answer + source; mutation tasks تحتاج state query أو DOM/backend check. كما نفصل browser prompt injection: نص الصفحة observation غير موثوق، لا يستطيع إعادة تعريف mission/authority. أي instruction من webpage تعامل كdata حتى يُثبت أنها جزء من task semantics.

Web Tool Gateway

Browser actions تمر Authority + domain allowlist + credential broker + effect class. External form submit أو delete ليس مجرد click؛ يصنف /external irreversible effect ويحتاج idempotency token إن أمكن.

Evidence

URL alone ليس evidence كافياً؛ نحفظ relevant state/result + captured_at + provenance.

Replay-safe

counterfactual replay يعيد DOM snapshots أو mock environment، ولا يعيد external POST الحقيقي.

هذه البنية ضرورية لاختبار outreach/business automation لاحقاً دون تحويل autonomy إلى click spam أو duplicate side effects.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. web functional validators. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. website drift وduplicate side effects. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. self-hosted tasks + idempotency. نختفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. BrowserAdapter + CredentialBroker. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

TheAgentCompany: الهارنس كعامل رقمي داخل مؤسسة مصغرة

WORKPLACE TASKS · WEB + CODE + COMMUNICATION · CONSEQUENT ACTIONS

TheAgentCompany يحاكي شركة برمجيات صغيرة مع مواقع وبيانات داخلية، ويطلب من agents browsing، coding، program execution والتواصل. الإصدار المنشور أظهر أن نسبة معتبرة من المهام الأصعب ظلت خارج قدرة baselines، ما يبين أن **integration friction** بين التطبيقات والسياقات أكبر من subtask quality. [B05]

من منظور MOUSSAIF، benchmark Mission OS: يجب حفظ artifacts، messaging outcomes، identity/role، project state، والdependencies بين tasks. نجاح node منفرد لا يكفي إذا workflow النهائي لم يغلق. لذلك نضيف Cross-App World Model يربط message، repo branch، entities: ticket، credential scope، calendar/event، document.

> **Workplace closure**
> Mission complete فقط إذا acceptance criteria موزعة على الأنظمة جميعاً PASS؛ مثال: code merged + test pass + message sent + issue state updated.

```
Cross-App Coherence = Satisfied Linked Preconditions / Total Cross-System Preconditions
```

هذا benchmark أيضاً يختبر authority inheritance: agent قد يكون مخولاً تعديل repo لكنه غير مخول إرسال external message، role-specific leases تمنع خلط «يمكنه تقنياً» مع «مسموح له».

تعميق سببي وإنتاجي

الآلية السببية المفترضة. cross-app workplace coherence. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. partial completion أو communication harm. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. linked criteria + per-system receipts. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Cross-App WorldModel. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

> **Promotion contract**
> كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

> **Evidence contract**
> كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

الكتب الهندسية: من patterns إلى production constraints

AI AGENT BOOKS · LLM ENGINEERING · RELIABILITY / COST / SCALE

الأوراق تكشف mechanisms، والـrepos تكشف implementation، والكتب الجيدة تجمع production concerns. في الجزء الثالث نستفيد خصوصاً من كتب agent/LLM engineering الحديثة التي تنظم monitoring، cost، evaluation، multi-agent communication، state/persistence، orchestration، memory، tools، وdeployment. وظيفتها ليست إثبات ادعاء جديد، بل **كشف failure modes التي لا تظهر في benchmark صغير**: observability، queues، migrations، secret handling، retries، rate limits، incident response.

مرجع هندسي	القيمة لهذا البحث	ترجمة عملية
Building Applications with AI Agents - O'Reilly, 2025	components, multi-agent, reliability/cost/scaling	production checklist + protocol boundaries
AI Agents in Action - Manning, 2025/26	workflows, tools, memory, agent patterns	HyperAbility patterns
LLM Engineer's Handbook, 2025	LLMOps, eval, monitoring	telemetry/eval plane
Designing Large Language Model Applications, 2025	system design around LLM apps	provider/context/application boundaries

قاعدة الاستخدام

لا ننقل "best practice" إلى constitution. نربط كل practice بفشل محتمل واختبار. مثال: queue/backpressure تظهر ضرورتها عندما multi-agent bursts تضرب provider quotas؛ قبل ذلك تبقى engineering precaution.

جمع الورقة + book + benchmark + repo يمنع bias مصدر واحد: research يمكن أن يكون أنيقاً لكنه لا يغطي operations؛ repo قد يكون ناجحاً دون causal ablations؛ benchmark قد يختزل distribution؛ الكتاب قد يجمع خبرة دون قياس. triangulation هو المنهج.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. production failure modes من engineering literature. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. cargo-cult best practices. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. load/rate-limit/incident drills. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. controls مرتبطة measurable triggers. العقد الأعلى ثابت: Context/Capability Routing → Active Difference → Mission Contract → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

من الورقة إلى البروتوكول: طبقة interoperability الموحدة

MCP · ACP · AG-UI · A2A · ATIF · INTERNAL EVENT CONTRACT

الأنظمة الحديثة تتحرك نحو protocols، لكن لكل protocol موضع. MCP يربط agent بالأدوات/الموارد؛ ACP يربط coding agent/editor client؛ AG-UI يهدف إلى agent-user frontend event interoperability؛ A2A إلى agent/agent boundaries؛ ATIF/trajectory formats إلى evaluation/training interchange. الخطأ أن نجعل أحدها «نظام التشغيل الداخلي».

AL RAZZI يحتاج internal canonical schemas ثم adapters. **Mission/Event/Tool/Evidence contracts لا تُعرَّف وفق MCP أو UI protocol**. adapter يحول outward/inward مع versioning. بذلك لو تغير protocol لا نفقد internal semantics.

```
Internal ToolIntent <-> MCP Adapter <-> MCP Server
Internal Command/Event <-> ACP Adapter <-> Editor
Internal Event/State <-> AG-UI Adapter <-> Desktop/Web UI
Normalized Trajectory <-> ATIF Adapter <-> Evaluator/Trainer
```

Protocol Firewall

كل incoming external object يمر schema validation + trust classification + authority mapping. Protocol compliance لا يمنح trust.

Risk	Benefits
semantic leakage: external protocol limitations تتسرب إلى core. الحل canonical richer model + lossy adapter warnings.	replaceability، ecosystem access، testability.

هذه الطبقة تجعل MOUSSAIFXT فعلاً cross-environment execution fabric بدلاً من collection of scripts.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. adapters تعزل protocols. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. protocol churn وlossy translation. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. conformance + round-trip + loss warnings. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. Protocol Firewall. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract	Evidence contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.	كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

checkpoints، Event sourcing، وdurable cognition

RESTART CONTINUITY · SNAPSHOT + EVENT TAIL · PROCESS HANDLES

الوكالة طويلة الأفق تنهار إذا كان terminal process هو الذاكرة. نحتاج checkpoints، durable Mission Runtime: state snapshot + event tail، عند process boundaries، artifact/evidence refs، child-agent registry، registry. هذا المبدأ حاضر بصيغ مختلفة في Codex threads، LangGraph durability، Prime daemon، وHermes sessions، ونرفعه هنا إلى contract موحد.

Checkpoint ليس copy عشوائياً للذاكرة؛ هو boundary معنوي: before court/commit، before irreversible effect، after DAG node، before mutation، plan accepted. عملية resume تعيد verification للحالة الخارجية؛ لا تفترض أن process/file/network بقي كما كان منذ checkpoint.

Reconciliation on resume

(1 load snapshot 2) replay events 3) inspect external fingerprints 4) detect drift 5) mark stale nodes/evidence 6) resume/replan هذا يمنع "resume into a different world."

Ephemeral	Durable
token stream, transient UI spinner, disposable intermediate prose unless promoted.	Mission Contract, DAG, gaps, decisions, artifact/evidence refs.

Restart Continuity = Missions Safely Resumed / Missions Interrupted

تعميق سببي وإنتاجي

الآلية السببية المفترضة. snapshot + event tail للديمومة. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. event divergence وworld drift. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. random crash + reconciliation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. RFD يجعل restart release gate. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract	Evidence contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.	كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Secure autonomy: project injection، browser injection، secrets، supply-chain

TRUST DOMAINS · CREDENTIAL BROKER · PLUGIN TRUST · TOOL TRUST

عندما يصبح harness قادراً على قراءة repo/web واستخدام tools، كل مصدر سياق يمكن أن يحمل تعليمات adversarial. لذلك نفصل trust domains: ModelTrust، ProjectTrust، RemoteHostTrust، WebContentTrust، MCPTrust، PluginTrust. ملف AGENTS.md في repo موثوق قد يكون project instruction؛ نص في dependency README أو webpage لا يرتقي تلقائياً إلى authority.

CredentialBroker يعطي tool scoped secret في وقت التنفيذ دون إدخاله في model context أو PluginManifest. logs يعلن permissions، لكن SecureExecutionPlane ينفذها. dependency/upstream update لا يُقبل تلقائياً في harness genome؛ يمر eval + regression/security + license + lockfile.

تهديد	فشل naïve harness	Control
Repo prompt injection	model يطيع نصاً في ملف ويغير mission	trust labels + instruction provenance
Web injection	page يأمر بتسريب data	page=data + domain/tool policy
Secret leakage	API key يدخل PEK/trace	secret refs + broker + redaction
Plugin supply chain	plugin يملك shell/network	manifest + sandbox + signature/hash
Self-evolution bypass	candidate يزيل guardrail	immutable constitution + promotion court

قاعدة القدرة/السلطة

Agent يصبح أذكى أو skill يصبح أفضل لا يعني أن authority يرتفع. Capability evolution وpermission evolution مساران منفصلان؛ الثاني يحتاج مصدر سلطة خارجي واضح.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. trust labels وsecret isolation. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. indirect injection/exfiltration. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. repo/web injection + log scans. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. TrustDomain وSecretRef canonical. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

Computer/Browser Use: من pixel agent إلى hybrid resource agent

API → CLI → DOM → ACCESSIBILITY → VISION · EFFECT VERIFICATION

Computer Use يصبح أكثر موثوقية عندما لا يختزل إلى screenshot+click. نفس المهمة قد تُحل عبر accessibility tree، DOM، CLI، API أو vision. **Hybrid Resource Mode** يختار representation الأقل غموضاً والأكثر قابلية للتحقق، ويستخدم vision عندما لا توجد بنية أدق. هذا امتداد طبيعي لـprocess، DOM resource، screen state resource: EVCF resource وfile resource كلها mounted context.

ComputerAction يحمل intent/target/expected_effect/risk/verifier. بعد كل action مهم، observation جديدة تحدث WorldModel. إذا فشل loop، expected transition لا يواصل clicks عمياء؛ RemainingGap يعاد فتحه، وقد يغير action modality. مثال: button visually disabled → DOM says disabled → world model updates prerequisite gap.

Action modality router

Score = reliability_history + semantic_fit + verification_strength - ambiguity - side_effect_risk - latency. API قد يفوز في data mutation؛ DOM في web form؛ UIA في native app؛ vision في canvas/unknown app.

```
Computer Use = Perception × Action Grounding × State Verification × Recovery; clicks are only one substrate.
```

هذا يجعل AL RAZZI قابلاً لاختبار production work على Windows لاحقاً دون رهن AGI-oriented architecture بـVLM واحد.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. hybrid action representation. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. اختيار modality غير مكافئ. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. API/DOM/UIA/vision comparison. نحتفظ أبسط baseline ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. ActionModel يتعلم reliability. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

EVCF/MCAS: السياق الافتراضي كـresource fabric لا memory trick

EXTENSIBLE VIRTUAL CONTEXT FABRIC · MOUSSAIF COGNITIVE ADDRESS SPACE

نطوّر EVCF V3 إلى MCAS: كل عنصر معرفي أو تشغيلي يمكن عنونته دون أن يكون resident في prompt. الموارد تشمل evidence، datasets، KB units، symbols، source files، processes، agent results، hypotheses، claims، وcontext model. computer/browser states يصبح **materialized working set** فقط.

هذه architecture تحتاج أكثر من indexes، typed objects، URI resolver: متعددة vector DB، pin/evict، compute-to-context، cache invalidation، page faults، provenance، policies، وsemantic search. process status واحد لا يستطيع التعامل مع access policy. SQL aggregation، exact symbol، أو world-effect receipt.

Tier	أمثلة	عملية
L0 Resident PEK	goal, active gap, selected evidence	model inference
L1 Hot cache	recent pages, current branch artifacts	fast rematerialization
L2 Project graph	repo/world/context genome	structural search
L3 HyperKnowledge	KB/skills/abilities/loops	capability retrieval
L4 Structured stores	SQLite/DuckDB/JSON/logs	query/compute
L5 External/delegated	web/APIs/agents/Drive	fetch/delegate with trust

Cognitive page fault

إذا reasoning يحتاج object غير runtime، resident يسجل ContextNeed وي materialize page ثم يكمل. لا يُسمح للنموذج أن يتصرف كأنه قرأ resource كاملة لمجرد وجود handle.

MCAS objective: maximize decision-relevant information per resident token while preserving reconstructability.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. typed virtual resources. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. URI duplication وstale aliases. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. identity/hash/lineage/invalidation tests. نختفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. MCAS ResourceIdentity. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

من EVCF إلى AGI-oriented resource vascularity

CONTEXT ECONOMICS · COMPUTE-TO-DATA · MCD ROUTING · PREFETCH

الابتكار ليس فقط أن «السياق أكبر». بل أن resources لها تكلفة/سرعة/freshness/authority مختلفة، وMCD يمكن أن يوزع attention/compute عليها. نسمي ذلك Resource-Vascular Context: تدفق معرفي حسب active difference لا حسب حجم corpus. high-risk claim قد يستدعي primary source + independent verifier؛ code edit صغير قد يكتفي symbol map + tests.

prefetch يستخدم Mission DAG لتوقع next-node context؛ pin يحمي acceptance criteria/failures؛ eviction يزيل low-value summaries؛ compute-to-data يقلل materialization؛ cache invalidation يربط source hash. كل قرار يسجل في PEKTrace حتى نستطيع لاحقاً معرفة هل failure سببه model أم missing context policy.

Key distinction

Long-context model يوسع L0 فقط. EVCF/MCAS يوسع address space ويضيف computation/typing/provenance. كلاهما مفيد، لكن الثاني يمنع architecture lock-in إلى context window بعينه.

Ablation

same model: long-context stuffing vs MCAS paging vs hybrid; measure success/cost/reconstruction.

Metric

UsefulPageRate، ContextFaultRate، TokensAvoided، StaleContextErrors.

هنا يتجسد مفهوم Computational Vascularity الداخلي: الحوسبة تُوجّه إلى difference/evidence bottleneck، لا تتوزع بالتساوي على كل ملفات المشروع.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. vascular context allocation عبر MCD. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. إهمال prerequisite منخفض salience. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. hidden-prerequisite adversarial tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. MCD heuristic قابل للمعايرة. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو تسجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظرية 1 - سيادة فضاء العنونة المعرفية CASS

THEORY T1 · COGNITIVE ADDRESS-SPACE SOVEREIGNTY

CASS — Cognitive Address-Space Sovereignty

الفرضية: عمومية harness لا تعتمد فقط على سعة model context، بل على امتلاك runtime فضاء معرفياً مستقلاً عن النموذج: موارد قابلة للعنونة، access control، provenance، paging وreconstruction. نسمي هذا «سيادة فضاء العنونة»: النموذج مستهلك transient، بينما system owns cognitive addresses.

Mechanism. MCAS URI + typed resource registry + materialization policy + PEKTrace. إذا تم تبديل model/provider، تبقى identities والعلاقات والحالة. إذا فُقد model session، يعاد بناء resident set من addresses.

CASS Index = Reconstructible Cognitive Objects / Cognitive Objects Required for Mission Continuity

Prediction

في مهام تتجاوز context أو تتطلب CASS architecture، restart يجب أن تخفض context-loss failures مقارنة transcript-centric agent بنفس model.

Falsifier

إذا agent transcript+long-context يحقق نفس performance restart/large-resource بنفس أو أقل cost/complexity عبر عدة domains، قيمة CASS كطبقة مستقلة تنخفض.

Protocol

CTXREF: canonical typed context-reference schema يمكن adapters نقله دون materializing content.

Tool proposal

MCAS Resolver + ContextPageFault Profiler.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. runtime يملك cognitive state خارج model. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. إعادة تسمية DB/filesystem فقط. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. cross-model restart + >window context. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. CASS يثبت value عبر semantics/provenance/access. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability → Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظرية 2 - الوكالة ذات الأثر المطبوع WETA

THEORY T2 · WORLD-EFFECT TYPED AGENCY

WETA — World-Effect Typed Agency

الفرضية: reliability ترتفع عندما يكون لكل action type «توقيع أثر» يحدد expected state transition وverifier المناسب، بدلاً من أن تكون tools generic calls. الوكالة ليست sequence calls؛ هي typed transitions بين حالات العالم.

ToolIntent يتضمن EffectClass وExpectedDelta؛ ToolObservation لا يساوي success؛ EffectVerifier يقيس ObservedDelta؛ mismatch يفتح gap. هذه theory توحد process launch، email send، browser state، database row، software diff في semantics واحدة.

WETA Reliability = Verified Expected Transitions / Effectful Intentions

Prediction

typed effect contracts تقلل false-success خصوصاً في web/computer/cross-app tasks حيث API success أو click لا يضمن outcome.

Falsifier

إذا verification layer لا تخفض false-success أو تضيف latency دون gains على real environments، يجب تبسيط granularity أو استخدامها risk-based فقط.

Protocol

+ WE-Receipt: before fingerprint + intended delta + after fingerprint + evidence refs verdict.

Tools

EffectSchemaCompiler، WorldDeltaInspector.

تعميق سببي وإنتاجي

الآلية السببية المقترضة. typed world-effect transitions. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. verifier overhead. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. risk-stratified verification ablation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. EffectClass يحدد verification depth. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Observation → Evidence → Remaining Gap → Closure/Recovery → Bounded Execution. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظرية 3 - سلالة تطور الهارنس EHL

THEORY T3 · EVOLUTIONARY HARNESS LINEAGE

EHL — Evolutionary Harness Lineage

الفرضية: self-improvement يصبح قابلاً للعلم فقط عندما كل mutation في harness له benchmark score، diff، hypothesis، parent، وrollback path. بدون lineage لا يمكن فصل التحسن من drift أو قياس transfer.

HarnessGenome يجمع references للـ context/router/model/agent/tool/verifier policies. CandidateGenome يولد عبر AVO-style variation أو trace mining؛ shadow replay ثم held-out benchmark ثم canary. promotion لا يزيل parent history.

```
Evolution Gain = HeldOutScore(candidate) - HeldOutScore(parent), conditioned on equal/normalized budget
```

Prediction

lineage + held-out promotion يخفض regressions مقارنة in-place prompt/skill edits ويجعل causal attribution أفضل.

Falsifier

إذا candidate lineage لا يحسن rollback/debugging/selection مقارنة versioned Git/config فقط، تُدمج النظرية في ordinary release engineering.

Protocol

EVO-CANDIDATE manifest: parent_ids, mutation_class, tests, benchmark, canary, authority.

Tool

HarnessGenomeDiff + EvolutionJournal.

تعميق سببي وإنتاجي

الآلية السببية المقترضة. evolution lineage experimental. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. metadata لا تمنع objective gaming. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. held-out + canary + rollback drill. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. HarnessGenome refs immutable artifacts. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظرية 4 - شبكة الادعاء الإبستيمي ECL

THEORY T4 · EPISTEMIC CLAIM LATTICE

ECL — Epistemic Claim Lattice

الفرضية: agents طويلة الأفق تحتاج تمثيل claims مستقل عن prose memory، بحيث evidence/counterevidence/freshness/scope تغير status بوضوح. هذا يمد سلم E0-E5 الداخلي إلى runtime dynamic claim graph.

Claim لا يصبح VERIFIED لأن عدة agents كرروه؛ يحتاج evidence type مناسب. contradiction لا يُحل بالsummarization بل يحتفظ بالمسارين حتى discriminating evidence. ECL يرتبط بـFSKE/MCAL في تصنيف مراتب الدليل، لكن هنا نحدده كـdata structure قابلة للاختبار.

Status	معنى runtime	Action
UNKNOWN	لا دليل كاف	retrieve/experiment
HYPOTHESIS	تفسير قابل للاختبار	predict + test
SUPPORTED	دليل جزئي	seek independent evidence
VERIFIED	criterion met ضمن scope	may drive action
CONTESTED	counterevidence قائم	QMCD preserve alternatives
STALE	source/world changed	refresh

Prediction

ECL يقلل propagation للclaims القديمة/المتناقضة في research/ops missions.

Falsifier

إذا claim graph maintenance أعلى كلفة من benefit ولا يخفض factual/action errors، نستخدمه فقط high-stakes/long-lived projects.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. claim graph مستقل عن prose memory. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. claim bloat/duplicates. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. contradiction + stale-source tests. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. HyperKB يربط claims بالمصادر. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظرية 5 - الإدراك متعدد الهارنس المحدود BPC

THEORY T5 · BOUNDED POLYHARNESS COGNITION

BPC — Bounded Polyharness Cognition

الفرضية: لا يوجد harness واحد يتفوق في كل substrates؛ يمكن رفع robustness بـfederation بين harnesses بشرط bounded delegation، canonical mission state، وstructured returns. القيمة تأتي من diversity دون state fragmentation.

MetaRouter يختار Native/Hermes/Prime/Codex/Claude/OpenCode حسب measured capabilities. external harness لا يرى كل MCAS؛ يحصل scoped PEK + authority lease. النتيجة تعود Result/Evidence/Artifacts/Gaps. delegation depth/budget يمنع recursive explosion.

```
Polyharness Gain = Best-Routed Verified Success - Best-Single-Harness Verified Success - CoordinationCost
```

Prediction

على mixed task suite، evidence-weighted harness routing يتفوق على fixed harness إذا performance profiles غير متطابقة عبر domains.

Falsifier

إذا coordination/context translation cost يزيل gain أو اختلاف semantics يرفع false-success، نحد federation إلى independent verifier/isolated tasks.

Protocol

/HAP - Harness Adapter Protocol: discover/health/capabilities/start/observe/cancel checkpoint/result.

Tool

HarnessCapabilityProfiler.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. complementary harness federation. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. coordination/translation overhead. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. routed vs fixed harness suite. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. MOUSSAIFXT federation وAL RAZZI mission control. العقد الأعلى ثابت: Mission Contract / Active Difference → Context → Observation → Evidence → Remaining Gap → Closure/Recovery → Capability Routing → Bounded Execution. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظريات 6-8: من skill causality إلى phase transition عند الركود

THEORIES T6-T8 · CSC · RVC · SISPT

T6 CSC · T7 RVC · T8 SISPT

T6 - Causal Skill Compilation (CSC)
Skill لا تُرقّى من repeated pattern فقط؛ يجب أن نثبت أن activation يسبب فرقاً إيجابياً عبر -matched/held-out tasks. Tool: SkillAblator. Protocol: SKILL EVIDENCE manifest. Prediction: يقلّ skill bloat ويرتفع transfer density.

T7 - Resource-Vascular Context (RVC)
السياق ليس buffer ثابتاً؛ موارد المعرفة/الحوسبة تتلقى bandwidth وفق :MCD: uncertainty, impact, evidence weakness, dependency centrality. Tool :ContextFlowProfiler. Prediction: نفس budget يحقق أعلى verified outcome من uniform retrieval.

T8 - Stagnation-Induced Search Phase Transition (SISPT)
عند plateau لا تزيد turns تدريجياً فقط؛ ننتقل «طورياً» من exploitation إلى :exploration: model/harness/representation/evaluator/hypothesis diversity. Tool: StagnationDetector + SearchPhaseController.

فاصل T8: transition لا يفعّل من boredom linguistic؛ يحتاج action-without-delta، score plateau، unchanged gap vector، signals: repeated failure signature، candidate diversity collapse. falsifier: إذا phase change يزيد cost دون حل stagnation عبر benchmark suite.

```
Search Phase = f(progress_slope, gap_entropy, repeated_failure, candidate_diversity, budget_remaining)
```

تعميق سببي وإنتاجي

الآلية السببية المفترضة. ربط skill/context/search بالقياس. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. تفاعل النظريات يصعب attribution. لذلك لا يعلن component نجاحه من داخله؛ يبقى closure خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. factorial ablation. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. لا meta-layer قبل إثبات primitives. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

النظريات 9-10: routing بالدليل وإعادة البناء أولاً

THEORIES T9-T10 · EWHR · RFD

T9 EWHR · T10 RFD

T9 - Evidence-Weighted Harness Routing (EWHR)

Router لا يعتمد model reputation أو static tags فقط. كل model/harness/tool/skill له PerformanceRegistry حسب task family: verified success, false success, cost, latency, recovery, court rejection. routing يوازن current capability fit مع evidence history.

```
RouteScore = Fit × Reliability × EvidenceStrength × Availability - Cost - Latency - Risk
```

T10 - Reconstruction-First Durability (RFD)

نصمم mission state بحيث fresh process يستطيع إعادة بناء العمل من artifacts/events/checkpoints قبل تحسين continuity UI. الحجة: long-horizon reliability = reconstructability، لا طول hidden context. Tool: ReconstructionTester.

Prediction T9: adaptive router يتفوق على fixed default في heterogeneous suite بعد warm-up كافٍ. **Falsifier:** history يتأخر/يخاز ويخفض performance، الحل contextual bandit أو conservative exploration، لا blind exploitation.

Prediction T10: kill/restart في random points لا يخفض completion بشكل كبير عندما durable state صحيح. **Falsifier:** state artifacts غير كافية لإعادة semantics أو stale world makes replay unsafe؛ عندها نضيف reconciliation لا مزيد logging فقط.

Protocol

mission/DAG/gaps/authority/artifacts/evidence/fingerprints /RECON-SNAPSHOT: event cursor.

Protocol

ROUTE-RECEIPT: alternatives + scores + chosen + reason + later outcome.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. القيمة لا تأتي من الاسم، بل من تحويلها إلى performance history + reconstruction laws. state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. cold-start/stale profiles. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. exploration+decay+restart benchmark. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. RouteReceipt وReconSnapshot. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

أدوات وبروتوكولات Part III + الربط المباشر بـAL RAZZI/MOUSSAIFXT

IMPLEMENTATION BLUEPRINT · 18 TOOLS · 9 PROTOCOLS · PRODUCTION TESTS

تحويل النظريات إلى engineering يتطلب artifacts محددة لا namespaces فارغة. نضع حزمة أدوات أولية قابلة للبناء داخل modules الحالية، وكل أداة يجب أن تُربط بـtest وtelemetry:

Tool	وظيفته	Protocol/Output
MCAS Resolver	resolve typed cognitive resources	CTXREF
ContextFlowProfiler	measure paging/pinning/utility	PEKTrace
WorldDeltaInspector	verify effect transitions	WE-Receipt
HarnessCapabilityProfiler	benchmark adapters	HAP profile
HarnessGenomeDiff	compare evolved harness versions	EVO-CANDIDATE
SkillAblator	test causal skill contribution	SKILL-EVIDENCE
StagnationDetector	detect plateau/failure loops	ProgressVector
SearchPhaseController	switch explore/exploit policy	PhaseDecision
RouteReceiptWriter	record routing alternatives/outcome	ROUTE-RECEIPT
ReconstructionTester	kill/resume randomized missions	RECON-SNAPSHOT
ClaimGraph	track evidence/status/freshness	ECL claim
ExperimentManager	hypothesis→test→observation	ExperimentReceipt
CredentialBroker	scoped secrets injection	SecretRef
EffectSchemaCompiler	derive verifier from tool contract	EffectSchema
HyperBranchManager	isolated code candidates	BranchReceipt
TrajectoryNormalizer	events→training/eval trajectory	ATIF-like record
GeneralizationCourt	promotion boundary for lessons	CourtResult
ChaosInjector	provider/tool/process/context failures	FailureReceipt

Production sequence: **P0 contracts** → P1 real provider/tool → P2 restart/recovery → P3 browser/computer → P4 MCAS stress → P5 cross-harness → P6 self-evolution shadow → P7 canary. لا ندخل self-evolution قبل أن تصبح trajectories والevaluation صلبة، وإلا سنطور الضجيج.

Minimum revolution gate

يجب أن يظهر تحسن مقاس مقابل baseline بسيط في 4 محاور على الأقل: recovery، context efficiency، restart continuity، false-success، verified completion، transfer. إذا لم يظهر، تُبسّط architecture حتى تتضح القيمة.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. أدوات محددة بدلاً من namespaces. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. architecture theatre عبر stubs. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. contract+runtime+test+telemetry+failure. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. breadth قبل vertical slice. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract
كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract
كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

برنامج الاختبار المقارن: كيف نثبت أن AL RAZZI أصبح أفضل فعلاً؟

BENCHMARK MATRIX · ABLATIONS · CHAOS · HELD-OUT TRANSFER · CANARY

نقترح benchmark program لا يختبر feature existence بل causal contribution. Baselines: A0 minimal ReAct/tool loop؛ A1 + repo/context map؛ A2 + HyperKB/PEK؛ A3 + WorldEffect; A4 + RemainingGap/ClosureCourt; A5 + MCAS؛ A6 + multi-harness routing؛ A7 + evolution plane. نفس model/budget قدر الإمكان، ثم نكرر على task families مختلفة.

Suite	الاختبار	المقياس الأساسي
SWE	real repo bug/feature + dirty workspace	verified patch + no-loss
Web	WebArena-like functional tasks	backend/DOM goal state
Computer	OSWorld subset/local tasks	execution-based success
Research	hypothesis/experiment/report	claim/evidence correctness
Knowledge	large corpus > model context	answer + provenance + tokens
Durability	random kill/restart	safe resume rate
Chaos	timeouts/provider/tool/db failures	recovery / false-success
Transfer	new task after learned skill/policy	held-out gain
Security	prompt injection/secret/tool bypass	violation rate

Production gate

لا تعلن Production-Grade لأن unit tests تمر. نطلب: provider failover، effect verifier، restart mid-execution، real-project mutation، real-provider E2E، at least one computer/browser flow، injection tests، وheld-out evolution gain.

Primary: VerifiedCompletion | FalseSuccess | MeanTimeToRecovery | RestartContinuity | UsefulContextRatio | CostPerVerifiedMission | TransferGain

الهدف ليس تعظيم autonomy duration. metric الأهم **verified progress per unit cost/time**. agent يعمل 12 ساعة بلا delta أسوأ من agent ينجز task في 25 دقيقة مع evidence.

تعميق سببي وإنتاجي

الآلية السببية المفترضة. benchmark matrix كاختبار causal architecture. القيمة لا تأتي من الاسم، بل من تحويلها إلى state transitions وأحداث قابلة للمراقبة: ما inputs، ما policy أو قرار routing، ما الموارد التي materialized، وما الأثر المتوقع؟ ثم يُربط ذلك بـevidence يمكن إعادة فحصها بعد الجلسة. هكذا يصبح اختلاف harnesses اختلافاً تجريبياً بدلاً من مقارنة واجهات أو claims تسويقية.

حد الفشل. comparisons غير عادلة بسبب budgets/env. لذلك لا يعلن component نجاحه من داخله؛ closure يبقى خارج producer ويستخدم WorldEffect أو Regression/No-Loss أو evaluator مناسب. الفشل المتكرر يسجل ErrorMemory مع scope ومصدره، ولا يتحول تلقائياً إلى قاعدة عامة أو memory دائمة.

اختبار حاسم. controlled budgets + repeated runs. نحتفظ baseline أبسط ونقيس cost وlatency وtokens وrecovery إلى جانب النتيجة. أي gain يختفي عند تغيير model أو task family يسجل كتخصص domain-specific، لا كدليل عمومية. وعند variance بين runs نحفظ التوزيع بدلاً من اختيار أفضل trajectory.

الدمج في MOUSSAIFXT. evidence rank يرتفع بعد runs محفوظة. العقد الأعلى ثابت: Mission Contract → Active Difference → Context/Capability Routing → Bounded Execution → Observation → Evidence → Remaining Gap → Closure/Recovery. هذا الخط يمنع feature جديدة من بناء loop موازية غير مرئية للحاكمية أو سجل الأدلة.

Promotion contract

كل lesson/policy/skill جديد يبدأ Candidate مع test مناسب وrollback pointer.

Evidence contract

كل claim تشغيلي يحتاج artifact/trace/state readback قابل لإعادة الفحص.

المراجع والعقد الختامي للبحث

PRIMARY SOURCES · INTERNAL CORPUS · FINAL RESEARCH CONTRACT

هذه القائمة تركز على المصادر التي غيرت معمارية Part III، لا على exhaustive bibliography. تواريخ/تفاصيل المشاريع سريعة التغير؛ عند implementation يجب pin commit SHA وlicense في upstreams.lock.json.

[E01] PrimeIntellect-ai/prime-agent — https://github.com/PrimeIntellect-ai/prime-agent
RLM, persistent IPython, subagents, daemon sessions, continual harness.

[E02] Prime Agent Continual Harness implementation — https://github.com/PrimeIntellect-ai/prime-agent/blob/main/packages/coding-agent/src/core/refinement/refinement.ts
Immutable base + editable persistent notes/memory/skills/subagents.

[E03] Chen et al. - AVO: Agentic Variation Operators for Autonomous Evolutionary Search (2026) — https://arxiv.org/abs/2603.24517
Agentic variation, lineage, execution feedback, long-horizon optimization.

[E05] NVIDIA OpenShell — https://github.com/NVIDIA/OpenShell
Sandboxed runtime, declarative filesystem/network/process/inference policies.

[E06] NVIDIA NeMo Agent Toolkit — https://github.com/NVIDIA/NeMo-Agent-Toolkit
Instrumentation, evaluation, trajectory/tool middleware.

[E07] Pydantic AI Harness — https://github.com/pydantic/pydantic-ai-harness
Capability packages: tools, hooks, instructions, model settings.

[E08] NousResearch/hermes-agent — https://github.com/NousResearch/hermes-agent
Persistent memory, skills, subagents, MCP, cron, provider independence.

[E09] NousResearch/hermes-agent-self-evolution — https://github.com/NousResearch/hermes-agent-self-evolution
DSPy/GEPA trace-driven skill/prompt evolution with gates.

[E10] OpenAI Codex — https://github.com/openai/codex
Core/app-server/CLI; sandbox and typed approval flows.

[E11] Anthropic Claude Code docs — https://docs.anthropic.com/en/docs/claude-code
Sessions, subagents, hooks, MCP, permissions, structured CLI.

[E12] OpenCode / Cline / Goose / Gemini CLI — https://github.com/anomalyco/opencode ; https://github.com/cline/cline ; https://github.com/block/goose ; https://github.com/google-gemini/gemini-cli
Portable CLI/headless/provider/protocol patterns.

[E13] Yang et al. - SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering — https://arxiv.org/abs/2405.15793
ACI design as a causal performance factor.

[E14] Aider Repository Map — https://aider.chat/docs/repomap.html
Graph-aware structural code context under token budget.

[E15] OpenHands / Deep Agents — https://github.com/All-Hands-AI/OpenHands ; https://github.com/langchain-ai/deepagents
Workspace/runtime abstraction, durable agent harness patterns.

[P01] Sumers et al. - Cognitive Architectures for Language Agents (CoALA) — https://arxiv.org/abs/2309.02427
Modular memory, structured action space, decision process.

[P02] Packer et al. - MemGPT: Towards LLMs as Operating Systems — https://arxiv.org/abs/2310.08560
Virtual context management and hierarchical memory.

[P03] Zhang et al. - Agentic Context Engineering — https://arxiv.org/abs/2510.04618
Evolving playbooks; generation/reflection/curation; anti-collapse.

[P04] Shinn et al. - Reflexion — https://arxiv.org/abs/2303.11366
Verbal feedback stored as episodic memory.

[P05] Park et al. - Generative Agents — https://arxiv.org/abs/2304.03442
Memory, reflection, planning architecture.

[P06] Wang et al. - Voyager — https://arxiv.org/abs/2305.16291
Automatic curriculum + executable skill library + feedback.

[P07] Microsoft Research - Agent Lightning — https://www.microsoft.com/en-us/research/project/agent-lightning/
Decouple agent execution from RL/optimization; trajectory interface.

[P08] Yamada et al. - The AI Scientist-v2 — https://arxiv.org/abs/2504.08066
Progressive agentic tree search for scientific discovery.

[P09] Schmidgall et al. - Agent Laboratory — https://arxiv.org/abs/2501.04227
Research workflow with human feedback.

[B01] Mialon et al. - GAIA — https://arxiv.org/abs/2311.12983
General assistants: tools/web/multimodal reasoning.

[B02] Liu et al. - AgentBench — https://arxiv.org/abs/2308.03688
Eight interactive environments and long-horizon failures.

[B03] OSWorld — https://os-world.github.io/
Real computer environment; execution-based evaluation; OSWorld 2.0 notice.

[B04] WebArena — https://webarena.dev/
Self-hosted realistic web environment with programmatic validators.

[B05] Xu et al. - TheAgentCompany — https://arxiv.org/abs/2412.14161
Workplace-like web/code/communication tasks.

المصادر الداخلية المستخدمة كقيود تصميم

MOUSSAIF AGI OS / MOUSSAIXT Deep Engineering Architecture, 50p. [M02] MOUSSAIF Unified AGI, 225p, claim-ranked E0-E5. [M03] [M01] AGI-Oriented Knowledge-Base as Cognitive OS + ParallaxWorld-MCD / Fattah-AGI corpus. تستخدم هنا لتثبيت vocabulary/invariants، لا كدليل خارجي على AGI.

العقد الختامي للجزء الثالث

لا يصبح AL RAZZI «ثورة» لأن هذه النظريات كُتبت، يصبح تقدماً حين: (1) يبني world effects حقيقية، (2) يستأنف بعد crash، (3) يعمل فوق context أكبر من model عبر MCAS، (4) يبدل harness/model دون state loss، (5) يتعلم candidate قابلاً للrollback، و(6) يثبت gain على held-out tasks مع انخفاض false-success. عندها فقط نرفع evidence rank من architecture إلى experiment.