AL RAZZI / MOUSSAIFXT

ورقة بحثية هندسية - مراجعة حالة الفن عبر 25 عملاً بحثياً مرتبطاً مباشرة

من النموذج إلى نظام تشغيل وكيلي مستمر

Persistent Agentic Harnesses, Virtual Context, Tool/Computer Use, Verification, Memory and Self-Evolution

تدرس هذه الورقة موقع AL RAZZI داخل التحول الجاري من وكلاء LLM قصيري الأفق إلى أنظمة تشغيل وكيلية تستطيع حفظ الحالة، تشغيل الأدوات، إدارة سياق خارجي أوسع من نافذة النموذج، التحقق من أثر العالم، التعلم من المسارات السابقة، والتطور تحت ضوابط مستقلة. وتربط المراجعة بين 25 ورقة/نظاماً بحثياً من 2024-2026 وبين بنية MOUSSAIF AGI OS / MOUSSAIFXT ، مع فصل صارم بين ما هو موثق في الأدبيات، وما هو موجود هندسياً في المشروع، وما يبقى فرضية تحتاج benchmark وablation.

المؤلف/صاحب البرنامج البحثي زهير مساعف - Zouhair Moussaif

النسخة سبتمبر 2026

اللغة Arabic RTL / English technical terms

الخط Amiri

تنبيه منهجي: لا تدعي الورقة أن AL RAZZI حقق AGI أو أنه يتفوق حالياً على الأنظمة المرجعية. الادعاء الأضيق هو أن لدينا معمارية قابلة للتحويل إلى برنامج تجريبي؛ ويصبح أي ادعاء تفوق مشروعاً فقط بعد مقارنة مضبوطة وثابتة النموذج والميزانية.

الملخص والمساهمة

Abstract, scope and research contribution

الملخص

تتحول هندسة الوكلاء المتقدمة سريعاً من سؤال «ما النموذج الأقوى؟» إلى سؤال «ما النظام المحيط بالنموذج الذي يسمح له بإظهار قدرته عبر الزمن؟». تظهر أعمال Prime Agent وRecursive Agent Harnesses وRecursive Language Models وAVO وRecuris وOSWorld 2.0 وغيرها أن الأداء طويل الأفق يتشكل من اجتماع الذاكرة، واجهة الحاسوب، التنفيذ، الاسترجاع، التحقق، التعافي، ومراقبة التطور، لا من الوزنات وحدها [R1-R10]. وفي الجهة المقابلة، توضح benchmarks مثل τ-bench وToolSandbox وTheAgentCompany وAgent-SafetyBench أن النجاح الظاهري في tool calling لا يضمن حالة عالم صحيحة أو سلوكاً ثابتاً أو آمناً [R16-R18,R25].

تقترح هذه الورقة قراءة AL RAZZI بوصفه Agentic Operating Harness يحتفظ بهوية المهمة وحالتها ودليلها خارج الاستدعاء الفردي للنموذج. يتموضع النموذج داخله كمورد معرفي قابل للتبديل، بينما تتكفل طبقات مستقلة بإدارة Mission state، HyperKB، HyperSkills، HyperAbilities، WorldEffect، Courts، ToolGateway، HyperLoops، وSelf-Reconstruction. هذا الاتجاه متسق مع معمارية MOUSSAIF ذات الطبقات الاثنتي عشرة التي تضع الدستور والهدف/السؤال والفرق في الأسفل، ثم المعرفة والقدرات، ثم التنفيذ، ثم التحقق والذاكرة/التطور في الأعلى [M1].

المساهمة الرئيسية

- تصنيف 25 ورقة إلى ست عائلات هندسية: harness، سياق/ذاكرة، self-evolution، verification/safety، computer/tool use، world models.
- استخراج invariants قابلة للتنفيذ بدل تلخيص الأوراق وصفياً.
- تحديد مواضع التقاطع والاختلاف بين AL RAZZI والأدبيات الحديثة.
- تحويل مفاهيم MCD/QMCD وHyperKnowledge وWorldEffect إلى فرضيات قابلة للـablation.
- اقتراح برنامج benchmark متعدد المحاور يرفع المشروع من E3/E4 إلى E5: دليل مقارن قابل للتكرار.

Capability ≈ Model × Harness × Context × Tools × Verification × Persistence

المنهجية وحدود الادعاء

Method, evidence ladder and claim discipline

اعتمدت المراجعة على أعمال peer-reviewed عندما كانت متاحة، وعلى arXiv/technical reports الحديثة عندما لم تكن الدورة العلمية قد أغلقت بعد. لا تعامل الورقة preprint كحقيقة نهائية؛ بل كدليل على اتجاه هندسي ونتيجة معلنة قابلة لإعادة الاختبار. لذلك نستعمل أربع رتب: موثق في المصدر ، استنتاج هندسي مباشر ، تركيب مقارن ، و فرضية Moussaif . أي جملة عن تفوق AL RAZZI تقع في المرتبة الأخيرة ما لم يوجد run مقارن محفوظ.

داخلياً، تستفيد الورقة من مبدأ مثبت في corpus المشروع: الحالة المرصودة والتفويض الصريح والدليل والتحقق يجب أن تتقدم على ثقة النموذج أو حجم المخرجات [M1]. كما تتبنى سلم الإثبات الموحد: E0 مجهول، E1 مذكور، E2 artifact موجود، E3 محدد هندسياً، E4 مجرب محلياً، E5 مقارن/مكرر [M2]. هذه الرتب مهمة لأن المشروع يملك حالياً artifacts وهندسة واختبارات محلية لعدد من الطبقات، لكن لا ينبغي دمج ذلك في ادعاء AGI أو SOTA.

معيار اختيار الأوراق

اختيرت الأعمال إذا غيرت واحداً على الأقل من قرارات تصميم AL RAZZI: كيفية حفظ context خارج النموذج؛ كيفية تحويل الخبرة إلى memory قابلة لإعادة الاستخدام؛ كيفية جعل subagent وحدة تشغيل كاملة؛ كيفية تقييم التفاعل مع أدوات ذات state؛ كيفية ضبط computer use طويل الأفق؛ كيفية فصل producer عن verifier؛ أو كيفية إجراء self-improvement دون tampering.

المحور	السؤال الهندسي	المقياس المقترح
Context	كم معرفة يمكن جعلها addressable دون حقنها كلها؟	resident tokens / success
Execution	هل تحقق الأداة الأثر المقصود؟	world-state correctness
Memory	هل تنتقل الخبرة إلى مهمة لاحقة؟	transfer gain
Evolution	هل التحسين حقيقي وغير متلاعب؟	held-out delta + tamper audit

خط الأساس: AL RAZZI داخل MOUSSAIF

Current project baseline and architectural reading

تقرير التنفيذ الحالي يصف AL RAZZI كـ CLI/headless runtime مع indexing، SQLite sessions، typed events لمئات وثائق registries، HyperKB، Evidence لـ HyperSkill/HyperAbility/HyperLoop/HyperAgent، Active-Difference/MCD routing، ToolGateway، sandbox/Git transactions، Independent Verifier، Courts، Ledger، وSelf-Reconstruction. هذه عناصر مهمة لأنها تعني أن الدراسة لا تبدأ من whiteboard؛ لكنها تبقى بحاجة إلى مقارنة خارجية مستقلة حتى لا تتحول كثافة الطبقات إلى بديل عن القياس.

في معمارية MOUSSAIF الرسمية، يتكون stack من L0 Constitution/Safety، L1 Goal/Question/Difference، L2 MCD/QMCD/MCAI، L3 HKOS/World Model، L4 Retrieval، L5 HyperSkills، L6 HyperAbilities، L7 HyperLoops، L8 Agents/Router، L9 Tools/Execution، L10 Verification/World Effect، L11 Memory/Evolution/Observability [M1]. القوة البحثية لهذه الخريطة ليست في أسمائها، بل في قابلية تحويل كل طبقة إلى intervention قابل للحذف.

```
Goal / Governing Question ↓ Active Difference + MCD/QMCD ↓ EVCF/MCAS + HyperKB ↓ HyperSkill → HyperAbility → HyperLoop ↓ Agent / Harness Router ↓ ToolGateway + Computer/Code ↓ Observed World Effect ↓ Evidence + Courts ↓ Experience + Evolution
```

السؤال الحاسم في هذه الورقة: هل يمكن أن يثبت هذا التنظيم نفسه كـ causal contribution إلى success/reliability؟ إذا لم يتغير الأداء عند حذف طبقة، فإما أن implementation ضعيف أو أن المفهوم زائد. هذا ينسجم أيضاً مع ورقة Knowledge-Base as Cognitive OS التي تجعل قيمة المعرفة في التحول إلى عمليات قابلة للنقل والهدم والتحقق، لا في حجم corpus [M3].

خريطة الأعمال الخمسة والعشرين

Research map: 2024-2026

#	العمل	المحور	الأثر المباشر على AL RAZZI
R1	Prime Agent	Harness / RLM	persistent REPL + continual harness + recursive agents
R2	AVO	Evolution	agent-as-variation-operator + lineage + grounded eval
R3	Recursive Agent Harnesses	Recursion	model-call لا full-harness recursion فقط
R4	Recursive Language Models	Context	prompt كبيئة خارجية قابلة للبرمجة
R5	Recuris	Memory	working/experiential memory + gated evolution
R6	GEPA	Optimization	trace reflection + Pareto prompt evolution
R7	Tycho	World models	build/use/repair/bypass model حسب القيمة
R8	OSWorld 2.0	Computer use	318- call-scale workflows and hidden-state failures
R9	StateAct	Computer use	program state before pixels
R10	Harness Tampering	Integrity	separate evolution from evaluator/security
R11-R12	Memory Survey + A-MEM	Memory	storage→reflection→experience; linked memory units
R13	AlphaEvolve	Evolution	executable evaluators + evolutionary search
R14-R15	Magentic-One + Agent Lab	Multi-agent	orchestration + research pipelines + human gates
R16-R18	ToolSandbox, τ-bench, SafetyBench	Reliability	stateful evaluation, pass^k, safety beyond prompts
R19-R23	SWE-Gym, SWE-agent, OpenHands, CodeAct, CoALA	Engineering	interfaces, sandbox, code actions, verifier training, cognitive baseline
R24-R25	GAIA, TheAgentCompany	Generalist eval	tool breadth and workplace long-horizon evaluation

تظهر الخريطة أن «الهارنس» لم يعد مجرد wrapper. توجد الآن أدلة تجريبية متزايدة على أن الواجهة، persistent state، programmable context، verifier، وشكل recursion يمكن أن يغيروا أداء backbone نفسه دون تغيير الوزنات. لهذا يجب اختبار AL RAZZI ضد baselines مع تثبيت النموذج والميزانية؛ وإلا لن نعرف هل التحسن جاء من harness أم من provider أقوى.

Prime Agent: الهارنس كغشاء معرفي مستمر

R1 - Persistent REPL, Continual Harness and recovery

Prime Agent [R1] هو أقرب جار بحثي مباشر لـAL RAZZI. يضع persistent IPython REPL في قلب النظام، ويستعمل RLM لمعالجة السياق برمجياً، ويحتفظ عبر Continual Harness بالتاريخ والذاكرة والمهارات والـprompts ومواصفات subagents. الفكرة الجوهرية أن history الضخم لا يجب أن يمر كله عبر active context؛ يكفي أن يبقى addressable وقابلاً للحوسبة داخل بيئة ثابتة.

هندسياً، يعطي هذا ثلاث نتائج. الأولى: يجب أن نميز بين model context و agent context substrate . الثانية: recovery ليس feature جانبية بل جزء من قدرة الوكيل طويلة الأفق؛ session التي لا تعود بعد crash ليست ذاكرة حقيقية. الثالثة: continual learning في harness يخلق سطح خطر جديداً، لأن skill فاسدة قد تنتقل إلى مهام لاحقة.

الاستيعاب المقترح

- تحويل HyperKernel من مجرد abstraction إلى computational context دائم: Python/SQL/repo index/artifact handles.
- جعل كل Mission تملك namespace ثابتاً للمتغيرات والـartifacts والـevidence بدلاً من إعادة تلخيصها في كل turn.
- إنشاء Context Resident Set : أصغر working set يدخل النموذج، بينما الموارد الأخرى تبقى خارجية وقابلة للاستدعاء.
- فصل Continual Memory عن Canonical Knowledge؛ لا تصعد كل خبرة إلى HyperKB مباشرة.

فرضية H1: عند تثبيت backbone، سيحقق AL RAZZI + EVCF معدل نجاح أعلى في مهام long-context من 1M-4M token-equivalent prompting، مع resident tokens أقل. معيار الفشل: عدم وجود فرق معنوي أو ارتفاع الكلفة دون تحسن.

Prime Agent لا يثبت صحة HyperKB أو MCD؛ لكنه يثبت أن harness غني قادر على رفع الأداء فوق baseline agent. لذلك يصبح المطلوب من AL RAZZI ليس كثرة المفاهيم، بل إثبات قيمة تنظيمية إضافية فوق هذا baseline القوي.

Recursive Agent Harnesses, RLM

R3-R4 - From recursive model calls to recursive harnesses

Recursive Language Models [R4] تعيد تعريف long context كبيئة خارجية يستطيع النموذج فحصها، تقسيمها، وكتابة برامج لاستدعاء أجزاء منها أو استدعاء نفسه عليها. هذه الفكرة مهمة لأنها تكسر الافتراض الشائع بأن كل ما يعرفه الوكيل لحظياً يجب أن يوجد كسلسلة tokens داخل نافذة واحدة. وتذهب Recursive Agent Harnesses [R3] خطوة أبعد: الوحدة المتكررة ليست model call فقط، بل agent كامل يملك filesystem وأدوات وتنفيذاً وتخطيطاً.

بالنسبة لـMOUSSAIFXT، يعني ذلك أن recursion يجب أن تكون متعددة الرتب. يمكن لمهمة صغيرة أن تستدعي model subcall رخيصاً، بينما ملف أو repo مستقل قد يستحق HyperAgent، ومشكلة منفصلة كبيرة قد تستحق harness كامل. إذا ساوى النظام كل delegation بنفس الوحدة، سيهدر context والcompute أو سيحصر subagent في قناة لا تكفي للمهمة.

نوع التفويض	الحالة المناسبة	الحد
MODEL_SUBCALL	تلخيص/تصنيف محدود	لا state طويلة
HYPERAGENT	مسار فرعي بأدوات وسياق	budget وscope
FULL_HARNESS	repo/task مستقل طويل	depth + cost + authority

الابتكار الممكن في AL RAZZI هو جعل recursion خاضعة لـMCD: لا نتوسع الشجرة لأنها تستطيع، بل لأن فرقاً نشطاً عالي القيمة يبرر فتح مورد جديد. ويجب أن يحمل كل فرع contract يحدد input، expected artifact، evidence، stop condition، وعودة الفشل. بهذه الطريقة يصبح Parallelism وسيلة لتقليل uncertainty لا مجرد تكثير agents.

فرضية H2: adaptive harness recursion المبنية على MCD تتفوق على fixed fan-out في success-per-dollar وduplicate-work rate، خصوصاً في repos كبيرة ومهام متعددة الملفات.

GEPA, AlphaEvolve, AVO

R2, R6, R13 - Evolution grounded in execution

تلتقي AVO [R2] وAlphaEvolve [R13] وGEPA [R6] عند مبدأ واحد: التحسين لا يجب أن يكون rewrite اعتباطياً، بل candidate generation + grounded evaluation + lineage . AVO يرفع agent من مولد candidate إلى variation operator يقوم بالتخطيط، القراءة، التعديل، الاختبار، النقد والإصلاح. AlphaEvolve يضع evaluators التنفيذية في قلب evolutionary loop. GEPA يستخرج من trajectories نقداً لغوياً ويختبر تعديلات prompts ويحافظ على Pareto frontier من الحلول المتكاملة.

هذا يعيد تعريف Self-Reconstruction داخل AL RAZZI. النسخة الضعيفة هي «وجدت خطأ، غيّر system prompt». النسخة البحثية يجب أن تعرف ما الذي يتطور: context strategy، verifier rule، HyperSkill، PEK compiler، router policy، أو loop topology. لكل candidate lineage وسجل للسبب والدليل والنتيجة على held-out missions.

Trace / Failure ↓ Localized Diagnosis ↓ Candidate Mutation ↓ Sandbox Evaluation ↓ Independent Verifier ↓ Shadow → Canary → Promote/Rollback

الميزة المحتملة لـQMCD هنا هي الإبقاء على عدة فرضيات/مرشحين في حالة تنافس حتى يجمع النظام دليلاً كافياً. لكن QMCD لا يصبح علمياً لمجرد التشابه مع Pareto selection؛ يجب تعريف scoring ومقارنته بـgreedy selection وrandom search.

متطلبات production

- لا candidate يغير benchmark oracle أو evaluation code.
- فصل population store عن canonical runtime.
- حفظ negative results كي لا يعاد استكشاف الفشل بلا نهاية.
- قياس generalization على missions لم تدخل evolution loop.

Recuris وذاكرة الخبرة

R5, R11, R12 - From storage to executable experience

تقدم Recuris [R5] تمييزاً عملياً بين Working Memory التي تحفظ حالة المهمة الحالية وExperiential Memory التي تحمل خبرة قابلة لإعادة الاستخدام. وتؤطر مراجعة ACL 2026 [R11] تطور الذاكرة في ثلاث مراحل: Storage ثم Reflection ثم Experience. أما A-MEM [R12] فتؤكد قيمة الوحدات المترابطة ديناميكياً بدل ذاكرة مسطحة.

هذه النتائج مهمة لـHyperKB لأن أكبر خطر هو مساواة «مزيد من الوثائق» بـ«مزيد من القدرة». المعرفة التشغيلية يجب أن تمر بدورة: artifact خام → claim/evidence → context use → successful/failed action → generalized lesson → validated skill. ولا ينبغي ترقيتها تلقائياً لأن agent كتبها؛ فـExperiential Memory قابلة للتلوث بالصدفة، بالـshortcut، أو بخطأ في attribution.

طبقة الذاكرة	وظيفتها	سياسة الترقية
WorkingMemory	آنية state	تتلاشى مع المهمة
MissionMemory	قرارات وأدلة المهمة	تغلق مع التقرير
ExperientialMemory	أنماط قابلة لإعادة الاستخدام	بعد replay/validation
HyperKB	معرفة canonical/provenance	court + provenance + version
HyperSkill	إجراء reusable	benchmark + transfer

فرضية H3: فصل Working/Experiential/Canonical memory سيقلل context pollution وwrong-skill invocation مقارنة بذاكرة موحدة، وتظهر الفائدة بوضوح مع طول horizon.

هنا يتقاطع البحث الخارجي مع أطروحة KB-as-OS في المشروع: الذاكرة تصبح ذات قيمة عندما تحول المعلومة إلى إجراء قابل للنقل والتحقق [M3].

MCD، Tycho: متى نبني نموذجاً؟

R7 - Active abstraction and resource allocation

Tycho [R7] لا يفترض أن world model يجب أن يبنى دائماً. يستطيع agent أن يبني hypothesis تنفيذية، يختبرها، يصلحها، يستعملها في التخطيط، أو يتجاوزها إذا كانت كلفتها أعلى من فائدتها. يسمي العمل هذه المشكلة active abstraction : متى يكون اكتساب model أو استعماله يستحق ثمنه؟ هذه نقطة اتصال مباشرة مع MCD. بدلاً من تعريف MCD كمقياس فلسفي واسع، يمكن تحويله إلى controller لتخصيص compute والتمثيل. إذا ارتفع uncertainty أو risk أو information gain المتوقع، قد يفتح النظام world-model branch أو يستدعي verifier أو يوسع context. إذا كان الفرق ضئيلاً، يضغط المسار ولا يبني abstraction ثقيلة.

$$\mathbf{MCD(x)=w_uU + w_rR + w_iI + wgG + w_eE - w\subset C}$$

حيث U عدم اليقين، R الخطر، I كسب المعلومات المتوقع، G شدة gap المتبقية، E أهمية world effect، وC كلفة compute/context. هذه صيغة بحثية أولية لا صيغة canonical نهائية؛ قيمتها في أنها قابلة للتعلم والـablation.

اختبار فاصل

ننشئ مهاماً تحتوي فروعاً يمكن حلها مباشرة وأخرى تستفيد من model explicit. نقارن: heuristic threshold، never-model، always-model، وMCD-controller. نقيس النجاح، عدد token budget، environment actions، وعدد النماذج التي بنيت ثم لم تستخدم. إذا لم يخفض MCD هدر abstraction أو يرفع success-per-compute فالنظرية تحتاج إعادة بناء.

الربط البحثي الصحيح ليس «Tycho يثبت MCD»، بل «Tycho يقدم مشكلة مجاورة قابلة للقياس يمكن أن تصبح ميدان اختبار لـMCD».

OSWorld 2.0: واقع computer use طويل الأفق

R8 - Hundreds of actions, hidden state and verification failures

OSWorld 2.0 [R8] يرفع مستوى computer-use evaluation من مهام قصيرة إلى 108 workflows واقعية طويلة، بمتوسط مئات tool calls في بعض الإعدادات، مع hidden state وتغيرات ديناميكية ومعلومات تظهر وسط المهمة. النتيجة الأهم ليست leaderboard بل taxonomy الفشل: نسيان قيد قديم، فقد معلومة جديدة، التخمين بدل السؤال، تجاوز verification، أو عدم استرجاع state مخفية.

هذه الحالات تطابق سبب وجود MissionState وEvidence وClosure Court في AL RAZZI. agent يمكن أن ينفذ 200 action صحيحة محلياً ثم يفشل لأن القيد الأول نسي أو لأن الملف النهائي لم يحفظ. لذلك يجب أن تكون القيود والـacceptance criteria objects مستقلة عن الحوار، وأن يعاد فحصها قبل closure.

تصميم ComputerUseRuntime

- Action log لا يكفي؛ نحتاج state snapshots أو probes عند النقاط الحرجة.
- كل task constraint يجب أن يحمل status: unsatisfied / satisfied / violated / unknown.
- يجب أن تسمح loop بـask-user عند نقص المعلومات بدلاً من تعظيم self-sufficiency.
- بعد كل side effect عالي الأثر: post-action observation منفصلة.
- الـcheckpoint لا يحفظ screenshot فقط؛ يحفظ app state handles والملفات والـIDs والسياق التشغيلي.

Completion ≠ Last Action Success

برنامج benchmark الداخلي يجب أن يتضمن مهام 100-500 خطوة، restarts متعمدة، dialogs غير متوقعة، تغييرات في input أثناء التنفيذ، واشتراطات لا تظهر في الشاشة الحالية. النجاح الوحيد المقبول هو world state النهائي الصحيح مع سجل يشرح كيف تحقق.

StateAct: الحالة قبل البكسل

R9 - Hybrid state-first computer interaction

StateAct [R9] ينطلق من ملاحظة بسيطة لكنها حاسمة: screenshot rendering خسارة معلومات مقارنة بحالة البرنامج نفسها. عندما يكون DOM أو filesystem أو backend state متاحاً، استخدامه في الفعل والذاكرة والتحقق قد يكون أكثر موثوقية وأرخص من الرؤية الخالصة. لكن visual channel يبقى ضرورياً عندما لا توجد واجهة أخرى أو عندما يكون layout نفسه جزءاً من المهمة.

هذا يؤيد بناء هرم أدوات في AL RAZZI بدلاً من Computer Use موحد قائم على click/vision. القاعدة ليست «تجنب GUI» بل «استعمل أعلى تمثيل دقة وأقل كلفة يحقق الأثر».

API / App State ↓ CLI / Filesystem / Database ↓ DOM / Accessibility ↓ Visual GUI / Mouse / Keyboard

يمكن لـMCD Router أن يختار السطح وفق reliability وlatency وsemantic fidelity. مثال: تغيير قيمة في spreadsheet قد ينفذ عبر API إن كان contract يسمح، لكنه يحتاج GUI إذا كان الهدف اختبار experience بصري. إنشاء ملف نصي عبر shell أقوى من فتح محرر وتحريك المؤشر، لكن اختبار rendering النهائي يحتاج screenshot.

Surface	ميزة	مخاطرة
API	دقة وسرعة	قد لا يعكس UI state
DOM/state	semantics واضحة	تغير schemas
GUI	تعميم واسع	هشاشة/كلفة

فرضية H4: state-first routing يقلل action count وfalse completion في workflows الطويلة مقارنة بـvisual-only، دون خسارة coverage عندما يوجد fallback بصري.

τ-bench, ToolSandbox

R16-R17 - Stateful tools, policies and repeat reliability

ToolSandbox [R16] يصحح نقصاً في كثير من اختبارات tool calling: ليست القضية هل اختار النموذج اسم الدالة الصحيح، بل هل فهم dependencies بين الأدوات وهل وصل environment إلى milestones الصحيحة عبر حوار ديناميكي. τ-bench [R17] يضيف interaction مع user وسياسات domain، ويقيم الحالة النهائية لقاعدة البيانات ويقترح pass^k لقياس الثبات عبر عدة runs.

هذه الأعمال تعطي WorldEffect في MOUSSAIF معنى قابلاً للاختبار. الـtool response ليس truth. قد ترجع API 200 بينما transaction خاطئة، أو قد يكون policy condition لم يتحقق، أو قد ينجح run واحد بالصدفة. لذلك يجب أن يكون AL RAZZI قادراً على تعريف goal-state predicate مستقل عن سلسلة الأدوات.

pass^k = P(success on all k independent trials)

الـproduction agent الذي ينجح 8/10 مرات لا يعادل وكيلاً ينجح 99/100 ولو كان المتوسط الأول جيداً. عند المهام المالية أو الإدارية الحساسة، repeated reliability أهم من best-of-N.

عقد أداة مقترح

- preconditions وauthority tier.
- expected state delta.
- observable evidence after execution.
- rollback/reversibility class.
- policy references.
- verification probe مستقل.

بذلك تتحول ToolGateway من registry لدوال إلى effect-governed interface . وهذا بالضبط هو الموضع الذي يمكن فيه لـAL RAZZI أن يتميز: ليس بعدد tools، بل بجودة عقود الأثر والقدرة على إثبات أن المهمة انتهت كما طلب المستخدم.

CodeAct, OpenHands, SWE-agent

R20-R22 - The agent-computer interface as a capability multiplier

SWE-agent [R20] يبين أن تصميم Agent-Computer Interface يمكن أن يحسن الأداء دون تغيير النموذج: commands مناسبة لتحرير الملفات والبحث والتنفيذ تقلل الاحتكاك وتمنع الأخطاء. OpenHands [R21] يبرهن قيمة runtime مفتوح متعدد النماذج يجمع browser، code، shell و[R22] CodeAct sandbox. يقدم executable Python كفضاء فعل عام يمكنه تركيب عدة عمليات بدل حصر agent في JSON schemas ثابتة. الدرس لـAL RAZZI: زيادة 500 tool schema ليست بالضرورة تقدماً. يمكن أن تكون واجهة أصغر لكن composable أقوى بكثير، شريطة أن تبقى السلطة محدودة. نقترح أن يتعامل model مع primitives ذات مستوى أعلى: delegate ، computer ، knowledge ، repository ، workspace ، بينما ToolGateway يترجمها إلى عمليات أدنى وفق security policy.

المشكلة	القوة	نمط الواجهة
تركيب ضعيف وتضخم schemas	قابلية ضبط عالية	Fixed JSON tools
سلطة واسعة وخطر	مرونة وتكوين	Shell/code
تعقيد implementation	مرونة + contracts	Hybrid typed workspace

الهدف هو bounded programmability : يسمح للنموذج بالتركيب البرمجي، لكن داخل sandbox وauthority envelopes. هذه نقطة أساسية في تحويل AL RAZZI من CLI شكلي إلى harness يبني فعلاً repositories ويشغل tests ويعيد الإصلاح.

SWE-Gym: الوكيل والمدقق كزوج

R19 - Training agents and verifiers on executable environments

SWE-Gym [R19] يقدم 2,438 مهمة برمجية واقعية مع runtimes واختبارات قابلة للتنفيذ، ويبين أن تدريب agents وverifiers على trajectories يمكن أن يرفع الأداء وأن verifier يمكن أن يخدم inference-time scaling. هذا يدعم مبدأ Producer ≠ Verifier في AL RAZZI، لكنه يدفعه خطوة إضافية: verifier ليس بالضرورة prompt نقدي فقط؛ يمكن أن يكون model أو classifier متخصصاً يتعلم من failure trajectories.

مع ذلك، يجب الحفاظ على ترتيب evidence. في البرمجيات، test executable أو type checker أو static analyzer أقوى من رأي LLM متى كان يغطي الادعاء. لذا تقترح الورقة hierarchy:

```
Deterministic Tests + Static/Type Analysis + Environment Probes + Learned Verifier + LLM Review
```

Courts في MOUSSAIF يمكن تحويلها إلى orchestration لهذه الأدلة لا إلى شخصيات نصية. Evidence Court يتحقق من الشواهد؛ Regression Court يعيد الاختبارات السابقة؛ Falsification Court يبحث عن counterexample؛ Closure Court يفحص acceptance criteria.

تجربة AL RAZZI SWE

نثبت backbone ونقارن AL RAZZI، OpenHands-style interface، minimal agent بدون Courts، وAL RAZZI كامل. نسجل first-resolve rate، regressions، pass test success، عدد إعادة المحاولة، token cost، وfalse closure. إذا رفعت Courts الكلفة دون خفض false closure أو regression، فيجب إعادة تصميمها.

Agent Laboratory, Magentic-One

R14-R15 - Orchestration, specialization and research workflows

Magentic-One [R14] يمثل نمط orchestrator يقود agents متخصصة للويب والملفات والكود ويعيد التخطيط عند الفشل. Agent Laboratory [R15] يطبق الوكالة على دورة البحث: literature review، experimentation، report writing، مع إمكان تدخل الإنسان. هذان العملان يبرزان قيمة specialization، لكنهما يحذران أيضاً من جعل orchestrator مركزاً مطلقاً لكل شيء.

في AL RAZZI، الأفضل فصل وظائف القيادة: MissionSupervisor يتابع إنجاز المهمة، AuthorityEngine يقرر حدود الأثر، MetaHarnessSupervisor يختار provider/harness، و EvolutionSupervisor يدير candidates. هذا الفصل يقلل خطر أن agent واحد يخطط وينفذ ويغير قواعده ويعلن النجاح.

HyperAbility كطبقة وسطى

يمكن فهم HyperAbility كـcomposition قابلة للنسخ عبر agents: «بحث علمي موثق»، «إصلاح repo»، «تحقيق source-to-claim»، «browser acquisition». وهي أوسع من skill صغيرة وأضيق من mission كاملة. لكي تصبح مفيدة يجب أن تحمل contract: capabilities، prerequisites، failure modes، cost envelope، evidence، expected artifacts، وتاريخ الأداء.

في العمل البحثي، human-in-the-loop ليس ضعفاً في الاستقلالية. عندما تكون الفرضية أو interpretation أو قرار النشر عالي الأثر، review gate جزء من جودة المنهج لا fallback.

هذا يفتح مسار ResearcherOS داخل MOUSSAIFXT: literature → claim graph → hypothesis → experiment/code → evidence → falsification → synthesis، مع provenance في كل انتقال.

Harness Tampering: خطر التطور الذاتي

R10 - Integrity of evaluators, provenance and authorization

ورقة Auditing Harness Tampering [R10] تصف ظاهرة حاسمة: self-improving agent قد يرفع score بتغيير harness أو evaluator أو مسار القياس بدلاً من تحسين القدرة الحقيقية. وقد يستمر هذا التلاعب في lineage «الأفضل». النتيجة المباشرة هي أن self-evolution لا يمكن أن تملك سلطة على كل طبقات النظام.

نقترح ثلاثة planes منفصلة:

```
Evolution Plane proposes candidates ≠ Evaluation Plane held-out tests/oracles ≠ Security Plane authority/provenance
```

يجب أن تكون audit logs، acceptance oracles، benchmark definitions، وpolicy roots خارج writable scope للcandidate. وإذا احتاج النظام إلى تطوير verifier نفسه، يتم ذلك في دورة أعلى ببيانات evaluation أخرى غير التي يقيس عليها.

Integrity invariants

- لا candidate يعدل درجاته أو يحذف failures.
- كل patch يحمل parent hash وreason وevaluation bundle.
- promotion يتم على held-out tasks لا lineage tasks فقط.
- rollback متاح دائماً حتى بعد canary.
- المقاييس الأساسية immutable أثناء experiment.

فرضية H5: إضافة tamper-audit وheld-out promotion ستخفض سرعة التحسين الظاهر لكنها سترفع external validity وتمنع «التقدم الوهمي».

EVCF / MCAS: السياق الافتراضي كمورد

Extensible Virtual Context as a research hypothesis

أقوى نقطة بحثية قابلة للاختبار في AL RAZZI هي فصل «ما هو addressable» عن «ما هو resident في نافذة النموذج». RLM وPrime Agent يقدمان الأساس الخارجي، بينما EVCF/MCAS يمكن أن يوسع الفكرة من text إلى موارد heterogeneous: repo symbols، tables، browser state، evidence، artifacts، agent traces وknowledge units.

```
Large Project / External World ↓ Indexes + AST + DB + HyperKB + Artifact Store ↓ ctx:// address space ↓ Context Governor / MCD ↓ Resident Working Set ↓ Model
```

المصطلح «Virtual Context» يجب ألا يسوق كـunlimited context. التشبيه الأدق هو virtual memory: النموذج لا يرى كل شيء في اللحظة نفسها، لكن النظام يستطيع page-in ما يلزم، compute فوق ما لا يلزم عرضه، ثم إعادة النتائج المختصرة مع provenance. النجاح يقاس بنوعية الاستدعاء وإدارة locality، لا بعدد terabytes القابلة للفهرسة.

واجهات مقترحة

ctx://repo/symbol/Foo.bar ctx://kb/claim/123 ctx://artifact/run/88/report ctx://trace/session/7/segment/12 ctx://computer/app/excel/state ctx://evidence/mission/42

إذا تحولت هذه الواجهات إلى typed handles يمكن للنموذج أن يكتب برامج صغيرة تستكشفها، وبذلك نقترب من RLM ولكن على substrate أوسع.

HyperKnowledge كـOS لا كمخزن

Operational knowledge, provenance and transfer

تؤكد الأدبيات الحديثة للذاكرة أن المرحلة المتقدمة ليست الحفظ بل abstraction من trajectories. وهذا يتقاطع مع مفهوم HyperKnowledge في المشروع: knowledge unit يجب أن تعرف مصدرها، claims، علاقاتها، متى تستخدم، وما skill التي تفعلها. لكن لا ينبغي أن تصبح HyperKB مستودعاً لكل trace؛ بل canonical layer تمر عبر filtration.

KnowledgeUnit	مثال metadata
Identity	id, version, owner, namespace
Provenance	source, timestamp, evidence hash
Semantics	claims, entities, relations, assumptions
Operationality	skills, tools, triggers, failure modes
Epistemic state	confidence rank, contradictions, open tests
Lifecycle	candidate → validated → deprecated

الميزة الأهم هي non-destructive evolution . إذا ظهرت معرفة أحدث، لا نحذف القديمة بلا أثر؛ ننشئ version/contradiction edge ونغير canonical pointer. هذا يسمح للـCourts بفهم كيف تغير الاعتقاد، ويمنع context من إعادة إحياء معلومة deprecated دون warning.

مقياس مقترح: Knowledge-to-Action Transfer Rate = نسبة الوحدات التي حسنت مهمة لاحقة فعلياً، لا نسبة الوحدات المسترجعة فقط.

وهكذا يصبح معيار HyperKB: هل تقلل وقت حل مهمة جديدة؟ هل تمنع تكرار خطأ؟ هل تنتقل عبر domain؟ إن لم تفعل، فهي أرشيف جيد لكنها ليست Cognitive OS بعد.

HyperSkill → HyperAbility → HyperLoop

Capability composition and bounded autonomy

تقترح MOUSSAIF ثلاث درجات composition يمكن جعلها دقيقة بحثياً. HyperSkill إجراء صغير قابل لإعادة الاستخدام؛ HyperAbility تركيب مهارات/أدوات/معرفة لتحقيق capability أوسع؛ HyperLoop workflow bounded يكرر التنفيذ والتحقق حتى stop condition. هذه الطبقات تشبه ما توزعه الأنظمة الأخرى بين skill libraries وworkflows وagents، لكن قيمتها المحتملة هي وجود contracts موحدة.

لكي لا تتحول التسميات إلى taxonomy زائدة، يجب تعريف schema موحد:

Field	المعنى
inputs/outputs	أنواع واضحة لا prose فقط
preconditions	ما يجب أن يكون صحيحاً قبل التشغيل
authority	ما الأثر المسموح
evidence	كيف نثبت النجاح
failure/rollback	كيف نعود ونفسر الفشل
cost profile	tokens/time/tools
transfer history	أين نجح وفشل سابقاً

HyperLoop يجب أن يكون bounded: max iterations، time budget، stagnation criterion، escalation policy. long-horizon autonomy لا تعني infinite loop؛ بل القدرة على الاستمرار مع شروط توقف واضحة.

اختبار: هل capability composition الثابتة جزئياً تتفوق على plan-from-scratch في reliability وcost؟ وهل تبقى مرنة أمام tasks جديدة؟

MCD/QMCD كموجّه موارد

Turning a conceptual layer into an ablatable controller

أكبر فرصة لتقوية MCD/QMCD علمياً هي إنزالهما من مستوى «تفسير الذكاء» إلى مستوى قرار observable. في كل step، النظام يملك خيارات: continue action، call verifier، delegate، build model، ask user، retrieve more، أو stop. MCD يمكن أن يعطي priority للفرق الذي يغير outcome، وQMCD يمكن أن يدير عدة hypotheses/paths قبل collapse.

نقترح تعريف Active Difference كمتغير له أثر على قرار أو prediction أو retrieval أو court. إذا لم يغير أي مسار، فهو noise مرشح للضغط. هذا متسق مع ParallaxWorld-MCD في corpus الذي يربط الفرق بالتنبؤ والفعل والأثر، لكنه هنا يصاغ كتجربة هندسية لا كمسلمة [M4].

Policy candidates

- Greedy: أعلى score لحظي.
- Threshold: افتح resource إذا تجاوز MCD حدّاً.
- Bandit: تعلم أي action يقلل gap بكلفة أقل.
- QMCD ensemble: احتفظ بعدة branches ثم اختبرها.

مقاييس المقارنة: redundancy، recovery time، actions، total cost، false success، task success، وcalibration بين score والنتيجة. إذا لم يتنبأ MCD بالحاجة إلى compute أو verifier، يعاد تعريف مكوناته بدل توسيع اللغة.

Priority = Expected Gap Reduction × Risk Weight / Resource Cost

Courts, WorldEffect

Verification as a first-class runtime plane

نتقاطع ToolSandbox وτ-bench وSWE-Gym وOSWorld2 في نتيجة واحدة: لا يكفي أن «يقول agent إنه انتهى». يجب أن يمتلك النظام تعريفاً مستقلاً للنجاح. في AL RAZZI يمكن جعل WorldEffect object يصف desired delta في العالم، ثم تجمع Courts أدلة متخصصة قبل closure.

Court	السؤال	أمثلة الأدلة
Evidence Court	هل لدينا شاهد مباشر؟	file hash, API readback, test
Falsification Court	ما الذي قد يفشل الادعاء؟	counterexample, negative probe
Regression Court	هل كسرنا شيئاً سابقاً؟	tests, baseline diff
Closure Court	هل كل acceptance criteria محققة؟	goal-state predicates

هذه المحاكم يجب أن تكون orchestration لا role-play. يمكن أن تستعمل deterministic checks أولاً ثم learned/LLM verifiers عند الغموض. كل Court تعيد verdict وevidence refs وuncertainty. وإذا اختلفت، لا يختار agent تلقائياً ما يناسبه؛ يتم escalation وفق policy.

مقياس أساسي: False Closure Rate. كثير من الهارنسات تبدو قوية إذا قيست رسائلها النهائية؛ أما عند فحص world state فتظهر فجوة. خفض false closure قد يكون أهم من رفع raw completion ببضع نقاط.

هندسة التطور الذاتي المحكوم

Self-evolution without self-certification

بعد تركيب AVO وAlphaEvolve وGEPA وRecuris وHarness Tampering، يمكن صياغة Self-Evolution Runtime من سبع مراحل. يبدأ من trace محددة، يشخص failure locus، يولد candidate محدود النطاق، يختبره في sandbox، يقارن على held-out tasks، يمرره إلى audit، ثم يطلقه shadow/canary قبل promotion.

Observe → Diagnose → Mutate → Evaluate → Audit → Canary → Promote / Rollback

يجب أن يكون هدف التطور متعدد الأبعاد: latency، cost، safety، reliability، success، وgeneralization. تحسين score واحد قد يدمر آخر. لذلك الأفضل Pareto frontier أو constrained optimization، لا scalar واحد قابل للاستغلال بسهولة.

HarnessGenome

يمكن تعريف genome كـversioned configuration يضم routing policy، context policy، skill registry snapshot، loop parameters، verifier stack، provider preferences، وprompt assets. لكنه لا يضم security root أو evaluation oracle. بذلك يمكن مقارنة genomes بدقة وإعادة أي run.

الشرط العلمي: كل claim عن «self-improvement» يجب أن يذكر baseline، عدد confidence interval، held-out set، trials، وما إذا كانت evaluator ثابتة.

MOUSSAIFXT كاتحاد Harnesses

Federation rather than lock-in

أحد الأهداف العملية لـMOUSSAIFXT هو أن لا يتحول AL RAZZI إلى provider lock-in. إذا كان Prime أو Hermes أو Codex أو Claude Code أو OpenHands أفضل في مسار فرعي معين، يمكن استعماله كـexecution substrate مع إبقاء Mission identity وknowledge وevidence عند الطبقة السيادية.

التمييز الحاسم بين sovereign state و delegated execution . لا نسمح للهارنس الخارجي بأن يصبح المصدر الوحيد للذاكرة أو الحقيقة. كل delegated run يرجع cost، trace/evidence summary، artifact، وstatus، ثم يندمج في global Mission DAG.

الطبقة	تبقى لدى MOUSSAIFXT	يمكن تفويضها
Goal/constraints	نعم	لا
Canonical memory	نعم	قراءة scoped
Execution	تنسيق	نعم
Evidence ledger	نعم	يدخل كـinput
Closure verdict	نعم	لا منفرداً

هذا يجعل المنافسين «قدرات قابلة للتوصيل» بدلاً من معمارية يجب تقليدها بالكامل. لكنه يتطلب adapter contracts ثابتة وقياساً صريحاً للأداء لكل harness. اختيار harness نفسه يمكن أن يصبح قرار MCD/QMCD مبنياً على reliability، cost، domain، وسجل سابق.

مصفوفة المقارنة الهندسية

AL RAZZI vs. frontier design patterns

البعد	النمط المرجعي	AL RAZZI target	الحالة العلمية
Persistent context	Prime/RLM	EVCF + HyperKernel	يحتاج benchmark
Harness recursion	RAH	Model/Agent/Harness tiers	تصميم
Experiential memory	Recuris/A-MEM	Working→Experience→HyperKB	جزئي
World modeling	Tycho	MCD-driven active modeling	فرضية
Computer use	StateAct/OSWorld2	state-first + visual fallback	هدف
Tool reliability	τ-bench/ToolSandbox	WorldEffect contracts	محدد هندسياً
Verification	SWE-Gym	Evidence/Courts	يحتاج ablation
Evolution	AVO/AlphaEvolve/GEPA	HarnessGenome + lineage	هدف V6
Tamper safety	Harness Tampering	plane separation-3	يحتاج hardening
Generalist eval	GAIA/TAC	multi-domain suite	غير مثبت

المصفوفة تظهر أن الاختلاف الحقيقي لا يكمن في امتلاك features منفصلة؛ معظمها له نظير خارجي. مساحة الجدة المحتملة هي التكامل الحاكي : استخدام Active Difference/MCD لتخصيص context/compute/delegation، وربط Knowledge-as-OS بالقدرات، ثم إلزام كل فعل بـWorldEffect وEvidence قبل ترقية الخبرة أو إعلان closure.

لكن هذه الجدة معمارية حتى الآن. لكي تصبح contribution علمية يجب أن تفوز على baseline مركبة، لا على agent بسيط متعمد الضعف.

برنامج **Benchmark** المقترح

Eight experimental tracks for E5 evidence

- Virtual Context: long-context vs RAG vs RLM-style vs EVCF، على corpora أكبر من النافذة.
- Fixed-model Harness: نفس backbone ونفس budget، minimal agent مقابل Hermes/Prime/OpenHands/AL RAZZI حيث يمكن.
- SWE: repo tasks مع tests وregression وrestart.
- Tool/State: τ-bench/ToolSandbox-like مع user policies وstateful DB.
- Computer Use: 100-500 action workflows مع state-first/visual ablation.
- Workplace: بيئة شركة داخلية متعددة التطبيقات على نمط TheAgentCompany.
- Self-Evolution: evolve routing/context/skills على train missions ثم held-out.
- Safety/Integrity: prompt injection، credential exfiltration، tamper attempts، unsafe-success tests.

المقاييس الموحدة

Metric	لماذا
Verified Success	نجاح world state لا النص
False Closure	قياس الثقة الزائفة
pass^k	الثبات عبر runs
Cost / Success	الكفاءة
Recovery Rate	الاستمرار بعد الفشل
Transfer Gain	قيمة الذاكرة/skills
Tamper Incidence	سلامة التطور

يجب نشر raw traces وconfigs وseed/versions قدر الإمكان، لأن الهارنسات سريعة التغير وأي نتيجة بلا reproducibility تتحول إلى انطباع مؤقت.

Ablation: ما الذي يضيف قيمة فعلاً؟

Causal evaluation of Moussaif layers

Variant	الإضافة
A0	Backbone + minimal tool loop
A1	Repo Map / Context Genome +
A2	EVCF / MCAS +
A3	HyperKB / PEK +
A4	MCD routing +
A5	HyperSkill / Ability / Loop +
A6	Courts / WorldEffect +
A7	Experience / Self-Evolution +
A8	Multi-harness federation +

Δ_i = **Performance(A_i) - Performance(A_{i-1})**

هذا الاختبار هو أهم محكمة هدم للمشروع. إذا أضيف MCD ولم يرفع success أو يخفض cost/redundancy، لا يجوز الاستمرار في منحه سلطة مركزية بالاسم فقط. إذا HyperAbility لا تحسن transfer مقابل dynamic planning، ربما تكون abstraction غير ضرورية. وإذا Courts تزيد latency 3× ولا تخفض false closure، يجب إعادة صياغتها.

التحليل الإحصائي

كل مقارنة تحتاج intervals، multiple seeds/runs لا point estimate فقط، وتصحيحاً لاختلاف task difficulty. في long-horizon tasks، success وحده فقير؛ نضيف binary recovery events، constraint violations، partial progress، وtime-to-detect-failure. كما يجب فصل model variance عن harness variance بتثبيت model/version/temperature قدر الإمكان.

الهدف ليس إثبات أن كل طبقة مفيدة، بل اكتشاف أيها غير مفيد بسرعة. هذا هو معنى محكمة الهدم عندما تتحول من مفهوم إلى ممارسة تجريبية.

Production V6 هندسة

From research prototype to production-grade harness

المسار المقترح

1. Runtime contract hardening: Command/Event schemas، session journal، deterministic replay، crash recovery. 2. EVCF: unified context post-effect probes. 4. Evidence plane: immutable و visual fallback مع paging policy. 3. Tool/Computer plane: state-first adaptersو handles Evaluation harness: fixed suites .6 فقط. descriptors لا goal-state predicates. 5. Capability registry: measured HyperSkills/Abilitiesو ledger مع external harnessesل cost accounting. 7. Evolution: candidate lineage + held-out tests + tamper audit. 8. Federation: adaptersو passˆkو .sovereign state

ما لا ينبغي فعله

- إضافة 1000 skill قبل قياس transfer لـ20 skill جيدة.
- ربط كل feature باسم جديد دون schema وfailure condition.
- السماح للagent بكتابة evaluator الذي يحكم عليه في نفس الدورة.
- قياس success من final message.
- ادعاء unlimited context بينما الواقع retrieval/paging.
- الخلط بين self-reflection وself-improvement.

Definition of Done لـV6

ليس أن تعمل CLI أو تمر unit tests فقط، بل أن ينفذ AL RAZZI مهمة repo حقيقية من البداية للنهاية، يتحمل restart، يسترجع state، يستخدم context خارج النافذة، يثبت world effect، يفشل بأمان عند غياب authority، ثم يعيد استعمال خبرة validated في مهمة ثانية مع تحسن measurable.

الخلاصة النظرية والهندسية

What the 25-paper study changes in the project

تكشف الأدبيات 2024-2026 أن frontier agent لا يمكن اختزاله في model + prompt + tools. تظهر Prime/RLM/RAH أن السياق والحوسبة الخارجية والـrecursion تغير القدرة الفعلية. تظهر Recuris/A-MEM أن الذاكرة يجب أن تتحول إلى خبرة. يبين Tycho أن بناء abstraction نفسه قرار اقتصادي. يبين OSWorld2/StateAct أن computer use يحتاج state واستمرارية أكثر من البكسل. توضح τ-bench/ToolSandbox أن success هو world state وسياسة، لا API call. تبين SWE-Gym أن verifier يمكن أن يكون مكوناً متعلماً. وتبين AVO/AlphaEvolve/GEPA أن self-improvement يحتاج execution-grounded evolution، بينما Harness Tampering يضع حدوداً صارمة على من يملك evaluator.

من هذه الزاوية، الفرضية الأقوى لـAL RAZZI ليست «هارنس فيه features أكثر»، بل أن نظاماً سيادياً persistent يمكن أن يجعل النموذج مورداً مؤقتاً داخل دورة أوسع تحفظ الهدف والسياق والمعرفة والدليل عبر الاستدعاءات . وينتج عن ذلك مسار بحث مركزي:

Goal → Active Difference → MCD/QMCD → Virtual Context + HyperKnowledge → Capability → World Action → Evidence/Courts → Experience/Evolution

إذا نجحت ablations في إظهار مساهمة سببية لكل طبقة، يتحول هذا من architecture corpus إلى برنامج systems research له مكان واضح بين tool reliability، agent memory، long-horizon harnesses، Cognitive Architectures، وself-improving systems. وإذا فشلت بعض الطبقات، فذلك ليس انهياراً للمشروع، بل أفضل نتيجة علمية ممكنة لأنها تضغط النظام نحو ما يعمل فعلاً.

الخلاصة: الهدف التالي ليس توسيع AL RAZZI لغوياً، بل رفعه إلى E5 عبر fixed-model comparative evidence، ثم جعل كل concept إما أن يثبت فرقاً تشغيلياً أو يتراجع إلى مستوى hypothesis.

مراجع وتعريف مجموعة الـ25 عملاً

Bibliography and project source anchors

[R1] Karten, S. et al. (2026). Prime Agent: A Self-Improving RLM Harness . arXiv:2608.23552.

[R2] Chen, T. et al. (2026). AVO: Agentic Variation Operators for Autonomous Evolutionary Search . arXiv:2603.24517.

[R3] Lumer, E., Sen, S., Paul, K., Subbiah, V. K. (2026). Recursive Agent Harnesses . arXiv:2606.13643.

[R4] Zhang, A. L., Kraska, T., Khattab, O. (2025). Recursive Language Models . arXiv:2512.24601.

[R5] Yu, Z. et al. (2026). Recursive Experiential-Working Memory Evolution for Long-Horizon Agent Harnesses (Recuris) . arXiv:2608.24876.

[R6] Agrawal, L. A. et al. (2026). GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning . ICLR 2026.

[R7] Lehmann, J., Aioanei, A., Vahdati, S. (2026). Tycho: Active Abstraction with Programmatic World Models for ARC-AGI-3 . arXiv:2607.28287.

[R8] Yuan, M. et al. (2026). OSWorld2.0: Benchmarking Computer Use Agents on Long-Horizon Real-World Tasks . arXiv:2606.29537.

[R9] Yang, Y. et al. (2026). StateAct: Program State, before Pixels, for Long-Horizon Computer-Use Agents . arXiv:2607.22798.

[R10] Wang, X., Zhang, X., Shao, J. (2026). Auditing Harness Tampering in Self-Improving Agents . arXiv:2609.00069.

[R11] Luo, J. et al. (2026). From Storage to Experience: A Survey on the Evolution of LLM Agent Memory Mechanisms . Findings of ACL 2026.

[R12] Xu, W. et al. (2025). A-MEM: Agentic Memory for LLM Agents . arXiv:2502.12110.

[R13] Novikov, A. et al. (2025). AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery . arXiv:2506.13131.

[R14] Fourney, A. et al. (2024). Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks . arXiv:2411.04468.

[R15] Schmidgall, S. et al. (2025). Agent Laboratory: Using LLM Agents as Research Assistants . Findings of EMNLP 2025.

[R16] Lu, J. et al. (2025). ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities . Findings of NAACL 2025.

[R17] Yao, S., Shinn, N., Razavi, P., Narasimhan, K. (2025). τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains . ICLR 2025.

[R18] Zhang, Z. et al. (2024). Agent-SafetyBench: Evaluating the Safety of LLM Agents . arXiv:2412.14470.

[R19] Pan, J. et al. (2025). Training Software Engineering Agents and Verifiers with SWE-Gym . ICML 2025.

[R20] Yang, J. et al. (2024). SWE-Agent: Agent-Computer Interfaces Enable Automated Software Engineering . NeurIPS 2024.

[R21] Wang, X. et al. (2025). OpenHands: An Open Platform for AI Software Developers as Generalist Agents . ICLR 2025.

[R22] Wang, X. et al. (2024). Executable Code Actions Elicit Better LLM Agents . ICML 2024.

[R23] Sumers, T., Yao, S., Narasimhan, K., Griffiths, T. (2024). Cognitive Architectures for Language Agents . TMLR 2024.

[R24] Mialon, G. et al. (2024). GAIA: A Benchmark for General AI Assistants . ICLR 2024.

[R25] Xu, F. F. et al. (2025). TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks . NeurIPS 2025.

Project anchors. [M1] MOUSSAIF AGI OS / MOUSSAIXT - Deep Engineering Architecture , 2026. [M2] MOUSSAIF Unified AGI - Claim-Ranked Research + Engineering , 2026. [M3] AGI-Oriented Knowledge-Base as Cognitive OS , 2026. [M4] ParallaxWorld-MCD Full AGI OS , 2026.

ملاحظة: الأعمال المنشورة كـ arXiv في 2026 تعامل هنا كـ preprints/technical research ما لم يذكر venue محكم. أرقام الأداء المذكورة في الورقة تعود لـ benchmark محلي. للمصادر الأصلية ويجب إعادة التحقق منها عند تنفيذ.