Beyond the Model: An Architectural Survey of Agent Harnesses

Anonymous ACL submission

Abstract

As LLM-based agents advance from static, single-turn question answering to long-horizon autonomous systems, their performance increasingly depends on the surrounding infrastructure, broadly referred to as the **Agent Harness**. Yet, the Agent Harness remains difficult to characterize, spanning heterogeneous mechanisms for execution, memory, and interaction. Existing surveys typically study these mechanisms in isolation, rather than as an integrated agent infrastructure. To bridge this gap, we introduce a **Unified Architectural Taxonomy** that organizes the Agent Harness into four interdependent layers: (1) Execution & Orchestration, (2) Context & Trajectory Management, (3) Interaction Surface & Execution Environment, and (4) Constraints & Guardrails. Using this taxonomy as an analytical lens, we examine how harness-level mechanisms shape agent capabilities across existing systems. This analysis yields two key findings: the Agent Harness is a widely overlooked performance bottleneck, and it co-evolves with the underlying models. These findings suggest that future harness architectures should be *built to delete*: designed with modular boundaries that allow obsolete scaffolding to be removed or replaced as model capabilities advance.

1 Introduction

Recent agent systems are increasingly moving beyond single-turn question answering to support long-horizon agentic tasks such as software engineering, web navigation, and data analysis. In these settings, performance depends not only on the intrinsic capabilities of the underlying models, but also on the surrounding infrastructure that manages tool use, maintains context, recovers from failures, and constrains interactions with external environments. This infrastructure, often implicit in system implementations, is what we refer to as the **Agent Harness**. Recent agent systems like OpenClaw (OpenClaw Contributors, 2026) and Claude

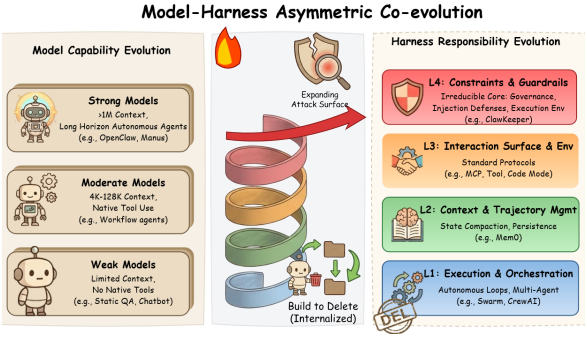

Figure 1: The Asymmetric Co-evolution of Model and Harness.

Code (Anthropic, 2025) exemplify this shift toward richer execution environments, where the harness must coordinate tool use, manage state, recover from failures, and enforce operational constraints.

However, existing surveys have largely examined LLM agents through isolated capabilities or modules, including reasoning and planning (Ferrag et al., 2025), memory and retrieval (Du et al., 2025a), tool use (Chowa et al., 2026), and multi-agent coordination, rather than as an integrated runtime harness that jointly governs execution, state, interaction with external environments, and operational constraints. As a result, the Agent Harness has not yet been systematically surveyed as a coherent architectural system, even though understanding its design trade-offs is essential for advancing effective agent systems.

To bridge this gap, we introduce a **Unified Architectural Taxonomy** that decomposes the Agent Harness by the runtime functions required to turn LLM outputs into agent behavior. It comprises four layers: (1) Execution & Orchestration, (2) Context & Trajectory Management, (3) Interaction Surface & Execution Environment, and (4) Constraints & Guardrails. The ordering reflects an outward expansion from internal control, to persistent state, to external actuation, and finally to system-level constraints. Using this taxonomy, we survey over 150 recent works spanning academic literature and

production systems to map the current landscape.

Our survey identifies several recurring **design patterns** in how agent harnesses are designed, evaluated, and deployed. First, harness design appears to be a major confounding factor in agent evaluation: differences in orchestration, memory management, tool interfaces, execution environments, and guardrails can materially affect observed performance, yet many benchmarks report results at the level of the combined model–harness system. This makes it difficult to disentangle model-intrinsic capabilities from infrastructure-level contributions. Second, the relationship between models and harnesses is dynamic rather than fixed. As models acquire capabilities such as longer context windows and native tool use, some previously external scaffolding may be simplified or absorbed; at the same time, more ambitious agentic applications introduce new infrastructure requirements, such as durable execution and monitoring. In particular, safety and guardrail mechanisms remain a persistent concern, as higher-autonomy systems expand the consequences of erroneous, adversarial, or unauthorized actions. Figure 1 summarizes this asymmetry: model improvements may absorb some scaffolding functions, while higher autonomy simultaneously increases demand for external constraints and governance.

Building on these findings, this survey makes three contributions: (1) a systematic definition and scope delineation of the Agent Harness; (2) a Unified Architectural Taxonomy organizing the Agent Harness into a four-layered stack (Section 2), with each layer analyzed through representative mechanisms, systems, and failure modes (Sections 3–6); and (3) a critical analysis of cross-harness evaluation and a roadmap distilled from the surveyed literature and systems arguing for "Built to Delete" harness architectures that can keep pace with the co-evolutionary cycle.

2 Taxonomy of the Agent Harness: A Unified Architectural Stack

Figure 2 illustrates the four-layer architectural abstraction of the Agent Harness and maps representative agent harness frameworks to their corresponding subfunctions. In this section, we discuss their applications and characteristics in more detail.

Layer 1: Execution and Orchestration. Layer 1 governs the control flow of an agent run: when to call the model, which tool or sub-agent to invoke next, how to pass intermediate results between steps, how to handle retries and failures, and when to stop. It focuses on the sequencing and coordination of actions, not on the content of memory, the semantics of tools, or the policies that restrict their use. Routing mechanisms select among available models, tools, or specialized agents according to task requirements, cost, latency, context length, or capability constraints. Multi-agent orchestration coordinates multiple agents or subagents by assigning roles, decomposing tasks, scheduling their execution, and merging intermediate results. This layer also includes execution-level resilience mechanisms, such as retry logic, backoff strategies, fallback routing, and recovery from failed model or tool calls. In our taxonomy, Layer 1 focuses on control flow and coordination, while leaving state representation to Layer 2, tool and environment semantics to Layer 3, and safety or guardrails enforcement to Layer 4.

Layer 2: Context and Trajectory Management. While the orchestration layer determines how execution proceeds, Layer 2 manages the information that persists across an agent run. It covers context construction, memory management, trajectory logging, state checkpointing, and retrieval. Its purpose is to keep relevant information available across model calls while controlling context length, reducing irrelevant or stale context, and preserving an auditable record of intermediate states and decisions. To address degradation caused by long, noisy, or poorly organized contexts (Wang et al., 2025d, 2026g), this layer uses context engineering techniques such as summarization, compression, retrieval, pruning, and structured memory updates. It also supports continuity across turns, tasks, and sessions through persistent memory stores, checkpoints, and state serialization.

Layer 3: Interaction Surface and Execution Environment. Since L(V)LMs operate primarily over tokenized inputs and outputs, Layer 3 defines the interfaces that allow an agent to observe, query, and act upon external environments, including tools, APIs, user interfaces, browsers, code interpreters, and execution sandboxes. This layer defines the agent's environmental footprint, standardizing interfaces for discrete tool calls and arbitrary code execution, effectively translating the abstract model's intent into concrete environmental state changes.

Layer 4: Constraints and Guardrails. Layer 4 defines the constraints under which an agent is

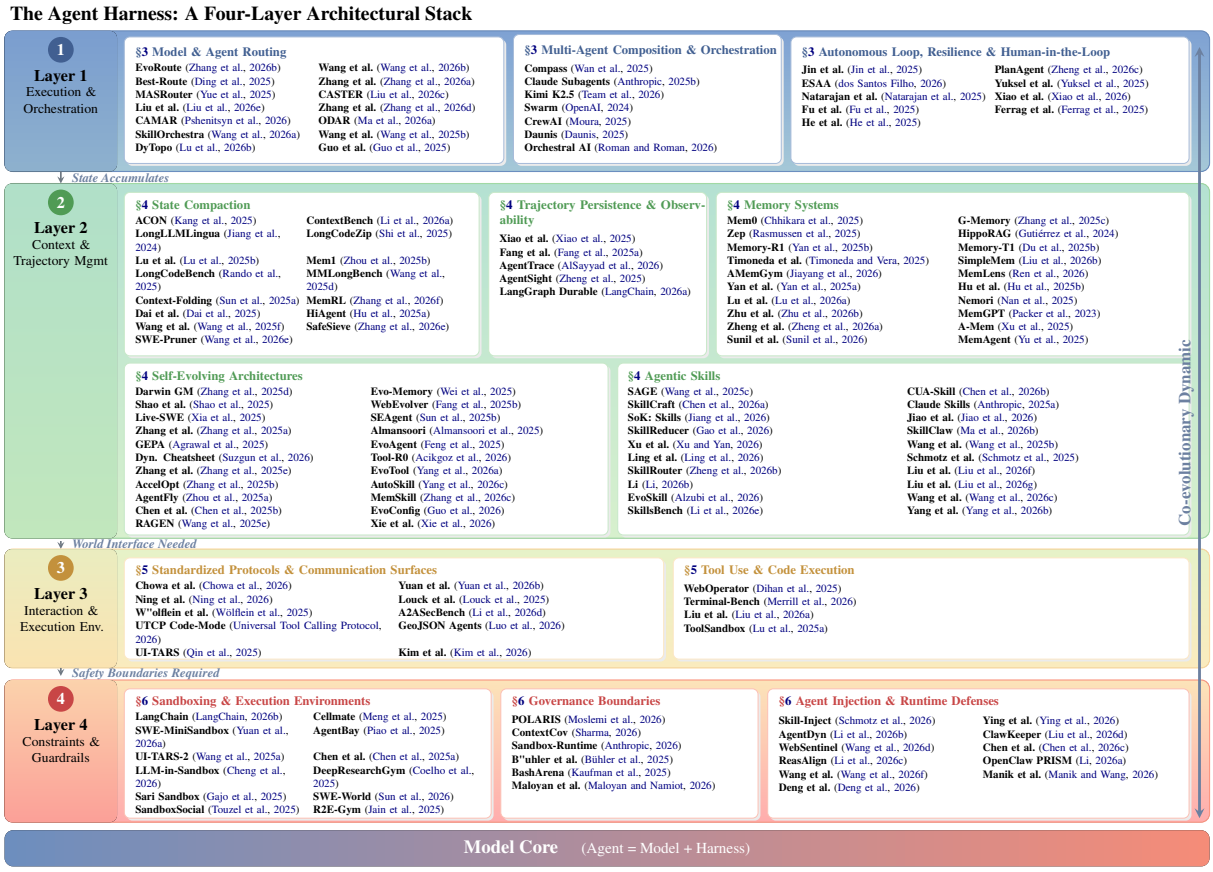

Figure 2: Taxonomy of the Agent Harness. The left column gives the functional role of each layer, the center panels group representative works by subfunction, and the vertical arrows indicate how execution expands outward from model control to world interaction and safety enforcement. Full classifications are provided in Appendix C.

allowed to act. It enforces deterministic **safety boundaries** on the system at two levels: *physical isolation* through process-, container-, VM-, or kernel-level sandboxing to limit the effects of unsafe agent behavior, and *logical governance* to validate inputs, outputs, tool calls, and execution results through schema checks, access-control rules, static analysis, runtime monitoring, or human review. This layer reduces risk by detecting or blocking unsafe actions before they affect external systems, and by constraining each tool invocation to the permissions required for the task.

Scope and Layer Boundaries. We place self-evolution and agentic skills in Layer 2 when they are represented as persistent, reusable state external to model weights. When such skills affect tool invocation or security, they also interact with Sections 5 and 6; we return to these cross-layer effects in those sections.

3 Layer 1: Execution & Orchestration

Layer 1 serves as the control plane of the Agent Harness: it runs the execution loop, routes subtasks to suitable models or agents, composes multi-agent workflows, and recovers from failures.

Model & Agent Routing. A central function of Layer 1 is *routing*: deciding which model, tool-using agent, or subagent should handle each subtask. Recent work can be grouped into three design patterns. First, *quality-oriented routing* selects agents by predicted competence or task fit, including experience-driven self-routing, learned model-agent selection, skill-transfer routing, semantic topology matching, and general model-agent orchestration (Zhang et al., 2026b; Wang et al., 2025b, 2026a; Lu et al., 2026b; Guo et al., 2025). Second, *cost- and budget-aware routing* trades accuracy against test-time compute, latency, or memory cost, as in Best-Route, CASTER, budget-aware routing, confidence-aware allocation, and query-aware memory-tier routing (Ding et al., 2025; Liu et al., 2026c; Zhang et al., 2026a; Wang et al., 2026b; Zhang et al., 2026d). Third, *domain- or modality-specific routing* adapts orchestration to multi-agent, VLM, continuous-action, or active-inference settings (Yue et al., 2025; Liu et al., 2026e; Pshenitsyn et al., 2026; Ma et al., 2026a).

These methods improve efficiency and specialization, but often assume reliable task-difficulty estimates, stable agent capabilities, and low routing overhead. Future harnesses therefore need calibrated routers that jointly optimize quality, cost, latency, and safety under distribution shift.

Multi-Agent Composition & Orchestration. Modern harnesses increasingly treat agents as **composable modules** rather than monolithic systems. Existing systems fall into three broad patterns. *Lightweight handoff frameworks*, such as OpenAI's Swarm, model collaboration as explicit function-call transfers between peer agents (OpenAI, 2024). *Hierarchical subagent architectures*, such as Anthropic's subagents (Anthropic, 2025b) and Kimi K2.5 (Team et al., 2026), let a parent agent spawn specialized child agents and aggregate their outputs. *Framework- or language-level orchestration* provides reusable abstractions for roles, protocols, and declarative workflows, as in CrewAI, Orchestral AI, and declarative agent workflow languages (Moura, 2025; Roman and Roman, 2026; Daunis, 2025). The Compass framework further addresses long-horizon coherence by maintaining an evolving shared context representation (Wan et al., 2025). The main trade-off is between flexibility and control: richer composition enables specialization and parallelism, but introduces coordination overhead, duplicated context, inconsistent state, and unclear accountability. A key direction is to make delegation, state handoff, and result aggregation explicit, inspectable, and recoverable.

Autonomous Loop, Resilience & Human-in-the-Loop. Beyond routing and composition, the autonomous loop must keep itself from stalling, drifting, or running indefinitely. Practical harnesses combine deterministic recovery primitives with event-sourced execution logs that support deterministic replay and branch-level recovery (Jin et al., 2025; dos Santos Filho, 2026), utilizing strategies such as structured retries, exponential backoff, error-classification-based rerouting, checkpointing, and fallback mechanisms. Human involvement spans a continuum: full human-in-the-loop oversight, intermittent guidance for test-time self-improvement, and fully closed-loop planning agents (Natarajan et al., 2025; Fu et al., 2025; He et al., 2025; Zheng et al., 2026c). These mechanisms form a **cross-layer feedback loop** that enables self-correcting workflows, where the harness detects, diagnoses, and recovers from failures without direct human intervention (Yuksel et al., 2025; Xiao et al., 2026). The resulting trade-off is autonomy versus verifiability: stronger closed-loop execution reduces supervision cost, but makes failure attribution, termination guarantees, and responsibility boundaries harder. As models gain stronger self-correction, Layer 1 shifts from compensating for model weaknesses toward coordinating capabilities that no single model can provide alone.

4 Layer 2: Context & Trajectory Management

Layer 2 governs the agent's information state: what remains in the active context, what is externalized, what is retrieved as memory, and how experience becomes reusable skills or self-improvement.

State Compaction. As trajectories grow, the fixed context window becomes a bottleneck: irrelevant history can accumulate and degrade reasoning, a failure mode we refer to as *context rot* (Wang et al., 2025d, 2026g). Broadly speaking, existing compaction methods fall into three patterns. *Prompt and token compression* removes low-utility tokens or domain-specific redundancy before inference (Jiang et al., 2024; Shi et al., 2025; Kang et al., 2025). *Agent-aware pruning* uses trajectory structure, repository context, or communication channels to discard task-irrelevant state (Wang et al., 2026e; Zhang et al., 2026e); however, the "Complexity Trap" result suggests that simple observation masking can rival expensive LLM-based summarization (Lindenbauer et al., 2025). *Context folding and hierarchical memory* restructure long histories into summaries, learned retention policies, or hierarchical working memories (Sun et al., 2025a; Lu et al., 2025b; Zhou et al., 2025b; Dai et al., 2025; Hu et al., 2025a). Evaluation is shifting accordingly from generic long-context QA to agent-specific retrieval and coding benchmarks (Li et al., 2026a; Zhu et al., 2026a; Rando et al., 2025; Wang et al., 2025d). The central trade-off is compression versus recoverability: aggressive compaction saves tokens but may erase the critical evidence needed for later verification, repair, or attribution.

Trajectory Persistence & Observability. Larger context windows (from 4K to 1M+ tokens) reduce immediate compression pressure but increase the need to persist, audit, and replay longer executions. Layer 2 therefore separates *working context* from *durable trajectory state*. One line of work com-

presses or checkpoints trajectories for reproducibility and recovery, including trajectory reduction, semantic checkpointing, notebook checkpointing, and durable execution runtimes (Xiao et al., 2025; Fang et al., 2025a; LangChain, 2026a). A complementary line externalizes **observability**: instead of storing logs inside the prompt, harnesses record tool calls, state mutations, and API events as structured traces for debugging, compliance, and performance analysis (AlSayyad et al., 2026; Zheng et al., 2025). The main tension is visibility versus overhead: richer traces improve diagnosis and governance, but increase storage cost, privacy risk, and the need for provenance-aware access control. As models gain native memory, the harness increasingly shifts from *providing* memory to *governing* it.

Memory Systems. Agent memory extends trajectory persistence into a structured, queryable knowledge layer (Huang et al., 2026), but existing systems differ in how they define and operationalize memory. Production-oriented platforms emphasize highly scalable extraction, retrieval, and temporal organization: Mem0 (Chhikara et al., 2025) builds long-term memory through automatic memory formation and retrieval, while Zep (Rasmussen et al., 2025) represents memory as a temporal knowledge graph. Research systems instead explore alternative management policies and cognitive abstractions. MemGPT (Packer et al., 2023) frames memory as an OS-like hierarchy that pages information between the context window and external storage; A-Mem (Xu et al., 2025) and Nemori (Nan et al., 2025) study self-organizing memories shaped by usage and cognitive principles; and HippoRAG (Gutiérrez et al., 2024) adapts hippocampal indexing theory for long-term retrieval. Recent RL-based approaches further learn memory policies directly: Memory-R1 (Yan et al., 2025b) trains agents to add, update, delete, and retrieve external memories, while Memory-T1 (Du et al., 2025b) learns time-aware evidence selection for temporal reasoning over multi-session dialogues. At the opposite end, SimpleMem (Liu et al., 2026b) shows that long-horizon memory requires no elaborate architectures, while broader agentic memory for deep-research agents is investigated in (Yan et al., 2025a).

Beyond single-agent recall, recent work increasingly treats memory as a coordination, adaptation, and governance substrate. In multi-agent settings, G-Memory (Zhang et al., 2025c) models hierarchical memories across agent teams, while MemAgent (Yu et al., 2025) uses RL-based memory agents to reshape long-context processing across conversations. Orthogonal work asks how a harness should choose and verify what to store: adaptive memory structures dynamically select storage strategies (Lu et al., 2026a), and provenance-aware tiered memory trades lossy compression for verifiable recall (Zhu et al., 2026b). Evaluation has likewise moved from static recall to interactional and task-level effects, including memory-assisted annotation performance (Timoneda and Vera, 2025), incremental multi-turn evaluation (Hu et al., 2025b), long-horizon conversational benchmarks such as AMemGym (Jiayang et al., 2026) and MemLens (Ren et al., 2026), and broader roadmaps for lifelong learning in LLM agents (Zheng et al., 2026a). These developments also expose memory as an attack surface: poisoning persistent memory can corrupt future retrievals and downstream decisions, as demonstrated for memory-augmented agents in (Sunil et al., 2026).

Self-Evolving Architectures. A defining aspiration of Layer 2 is enabling agents to **improve over their operational lifetime** without retraining the underlying model. In this broad sense, self-evolution covers post-deployment changes to the agent scaffold, context construction, world model, memory and retrieval policies, tool-use behavior, and multi-agent configuration. Existing work can be organized by the object of adaptation. *Architecture- and representation-level adaptation* includes architectural self-modification, live self-evolution for software engineering, continual world models, and agentic context engineering (Zhang et al., 2025d; Xia et al., 2025; Feng et al., 2025; Zhang et al., 2025e). *Policy- and strategy-level adaptation* refines memory, retrieval, prompts, and test-time strategies through RL or reflective optimization (Zhang et al., 2026f; Wang et al., 2025e; Agrawal et al., 2025; Suzgun et al., 2026). Other work expands self-evolution to specialized, operational, and collective settings. *Domain-specific adaptation* has been demonstrated for ML library development, accelerator-kernel optimization, and clinical interaction (Zhang et al., 2025a,b; Almansoori et al., 2025); AgentFly further shows that agent behavior can be tuned without fine-tuning the underlying LLM (Zhou et al., 2025a). *Tool- and memory-centric adaptation* evolves tool-use policies and reusable memory skills (Acikgoz et al.,

2026; Yang et al., 2026a; Zhang et al., 2026c; Wei et al., 2025), while *co-evolutionary systems* extend adaptation across agent populations and environments (Chen et al., 2025b; Fang et al., 2025b; Sun et al., 2025b; Guo et al., 2026). Yet self-evolution also creates failure modes: search-augmented agents may over-search (Xie et al., 2026), and unconstrained self-modification can induce emergent misevolution (Shao et al., 2025). These risks suggest that self-evolving architectures require not only adaptation mechanisms, but also explicit constraints, rollback, and auditability.

Agentic Skills. Agentic skills are reusable behavioral modules that extend an agent without changing model weights. They sit between raw model capability and external tools: unlike simple tool calls, skills encode procedures, execution policies, and task strategies (Jiang et al., 2026; Xu and Yan, 2026). Existing work can be organized around four core questions: how skills are acquired, selected, optimized, and evaluated. *Acquisition* methods study how agents learn skills through practice, experience, self-evolution, multi-agent discovery, or program induction (Chen et al., 2026a; Yang et al., 2026c; Alzubi et al., 2026; Wang et al., 2025f; Ma et al., 2026b). *Selection* and routing methods address how agents choose among large skill libraries, for example through retrieve-and-rerank mechanisms (Zheng et al., 2026b). *Optimization* focuses on making skills more efficient or compositional, including token-efficient skill reduction, skill-augmented generation, computer-use skills, and compositional skill synthesis for reasoning (Gao et al., 2026; Wang et al., 2025c; Chen et al., 2026b; Jiao et al., 2026). Finally, evaluation work examines both the architectural limits of skill-augmented single agents relative to multi-agent systems (Li, 2026b) and benchmark-based measurement of skill effectiveness across diverse tasks (Li et al., 2026e).

5 Layer 3: Interaction Surface & Execution Environment

Layer 3 defines the interfaces through which agents invoke tools, call APIs, and operate interactive environments such as terminals, browsers, GUIs, and code runtimes. It standardizes the boundary between model outputs and external actions.

Standardized Protocols & Communication Surfaces. A central Layer-3 problem is **interface standardization**: how should the harness expose tools, APIs, environments, and other agents to the model? One design pattern is *dynamic tool binding*, exemplified by delayed-discovery protocols such as the Model Context Protocol (MCP) (Maloyan and Namiot, 2026), which bind external APIs at run time and use progressive disclosure to avoid saturating the initial context. A second pattern is *structured code or plug-in interfaces*, such as UTCP's Code-Mode, which expose capabilities through more explicit executable interfaces (Universal Tool Calling Protocol, 2026). A third pattern is *agent-to-agent communication*, which seeks to move multi-agent systems beyond ad hoc message passing toward semantically aligned communication protocols (Yuan et al., 2026b). Google's A2A protocol is an early step in this direction, and protocol-aware benchmarks such as A2ASecBench indicate that agent-to-agent systems require security evaluation at the communication-protocol level (Li et al., 2026d). Across these designs, the main trade-off is standardization versus attack surface: richer protocols improve interoperability and late binding, but they also introduce new injection, data-exposure, and confused-deputy risks, as shown in recent analyses of MCP and A2A security (Louck et al., 2025).

Tool Use & Code Execution. The **tool-use interface** determines how agents discover, select, invoke, validate, and sometimes create functional capabilities (Chowa et al., 2026). A notable trend is *agent-driven tool creation*: agents can autonomously synthesize new tools when existing ones are insufficient (Wölflein et al., 2025). Tool reliability is equally critical; methods for detecting defects in LLM-based agent tools before deployment are proposed in (Ning et al., 2026). For agents operating under resource constraints, intention-based planning that optimizes tool invocation sequences under explicit budget limits is introduced in (Liu et al., 2026a). ToolSandbox (Lu et al., 2025a) provides a stateful, conversational benchmark for evaluating tool-use capabilities in realistic multi-turn settings, and agent robustness when APIs deviate from their idealized specifications is evaluated in (Kim et al., 2026). While functional APIs provide structured capabilities, many agentic tasks require more open-ended interaction through terminals, REPLs, GUIs, or browsers. Recent benchmarks and systems highlight this shift: Terminal-Bench evaluates realistic command-line

tasks requiring stateful interaction (Merrill et al., 2026), UI-TARS and WebOperator study native GUI and web control (Qin et al., 2025; Dihan et al., 2025), and geospatial agents expose trade-offs between function calling and code generation (Luo et al., 2026). These diverse interaction surfaces underscore the need for Layer 3 to provide a *unified abstraction* over heterogeneous execution environments. Indeed, as models gain native function-calling and code-generation capabilities, complex, OpenClaw-class applications demand richer protocol support, multimodal interfaces, and tighter integration with the sandboxing and permission mechanisms discussed in Layer 4. The evolution from single tool calls to full terminal and GUI execution illustrates this clearly: each advance in model capability unlocks more ambitious applications, which in turn require more sophisticated execution environments.

6 Layer 4: Constraints & Guardrails

Layer 4 provides isolation, permission control, and attack defenses that operate outside the model itself, so that unsafe or adversarial outputs do not directly translate into harmful actions.

Sandboxing & Isolation. Sandboxing constrains where agent actions execute, what resources they may access, and how side effects are contained. Existing designs follow two basic patterns: *agent-in-sandbox*, where the agent itself runs inside a constrained environment, and *sandbox-as-tool*, where the agent invokes isolated environments as external resources (LangChain, 2026b). Recent systems instantiate this idea along three directions. First, *execution-containment sandboxes* isolate code, filesystem, network, and process-level effects for production or software-engineering agents (Anthropic, 2026; Yuan et al., 2026a; Sun et al., 2026; Jain et al., 2025). Second, *task-environment sandboxes* reproduce realistic web, research, retail, social, or browser settings so that agents can be evaluated under controlled but stateful conditions (Meng et al., 2025; Coelho et al., 2025; Gajo et al., 2025; Touzel et al., 2025). Third, *hybrid interaction sandboxes* support seamless human intervention and environmental grounding, reflecting the fact that many deployments require mixed human–AI control rather than full autonomy (Piao et al., 2025; Cheng et al., 2026). The main limitation is coverage: static sandboxes improve reproducibility and containment, but they often underapproximate open-ended social, economic, and adversarial dynamics. This motivates lightweight, procedural, and co-evolutionary environments that can scale beyond fixed testbeds while retaining inspectable boundaries (Chen et al., 2025a).

Governance Boundaries. Because model outputs do not constitute enforceable safety guarantees, Layer 4 constrains agent actions through external **governance boundaries**, combining pre-execution validation of action structure with runtime containment of execution authority. Pre-execution mechanisms include schema-constrained decoding, static code analysis, and plan validation. Recent systems differ mainly in the source of enforceable constraints: POLARIS (Moslemi et al., 2026) validates each action against a formal type system before dispatch, whereas ContextCov (Sharma, 2026) compiles natural-language instruction into executable coverage constraints, reducing reliance on manual review. Complementarily, runtime governance addresses privileges. Sandboxed and lifecycle frameworks (Anthropic, 2026; Bühler et al., 2025) enforce permission boundaries to prevent local errors from amplifying into system-level failures under high-privilege execution (Kaufman et al., 2025). However, these mechanisms struggle with *transient authority*: agents often need short-lived access for specific sub-tasks, contradicting conventional persistent access models. This motivates intent-aware, least-privilege harnesses with ephemeral permissions, revisited in Appendix B.

Agent Injection, Persistent Threats, & Runtime Defenses. The expanded interaction surface of agentic systems introduces **agent injection**: adversarial content in tools, retrieved pages, plugins, or execution environments manipulates agent behavior. Existing attacks can be grouped by persistence and propagation. *Indirect injections* exploit web pages, documents, or tool outputs that agents treat as observations rather than instructions; defenses therefore require detection and localization at runtime, not only input filtering (Wang et al., 2026d; Li et al., 2026c). *Extension-level injections* target reusable skills and plugins, turning one-shot prompt manipulation into persistent attack surfaces through malicious skills, hidden comments, or vulnerable skill repositories (Schmotz et al., 2025, 2026; Wang et al., 2026c; Liu et al., 2026g,f). *Self-reinforcing injections* further exploit memory or self-evolution loops, producing "zombie agents" that preserve adversarial control across update cy-

cles (Yang et al., 2026b). Dynamic benchmarks such as AgentDyn show that these attacks should be evaluated as evolving adversarial processes rather than fixed prompt templates (Li et al., 2026b). **Defenses** similarly follow a layered pattern. Platform-specific systems such as ClawKeeper and OpenClaw PRISM combine skill validation, plugin sandboxing, runtime watchers, and defense-in-depth execution controls, while trajectory-based auditing inspects execution traces for anomalous behavior after the fact (Liu et al., 2026d; Li, 2026a; Chen et al., 2026c). Offensive studies of personalized local agents and OpenClaw-style ecosystems illustrate that the core tension is not platform-specific: any harness that admits user-authored behavioral extensions must balance extensibility against persistent compromise (Wang et al., 2026f; Manik and Wang, 2026). Broader threat taxonomies further show that injection interacts with data exfiltration, tool misuse, lateral movement, and multi-step deception across the full autonomous-agent attack surface (Deng et al., 2026; Ying et al., 2026). These results motivate that tool access, environmental state, and execution privileges should be scoped to declared task intent, observable execution context and revoked upon completion or deviation.

7 Future Research Directions

Across the four layers surveyed, a recurring pattern emerges: the Agent Harness is not a static scaffold, but an adaptive system boundary that co-evolves with the underlying model. This co-evolution creates asymmetric design pressure across layers. For Layers 1–3, stronger models may partially absorb functions previously implemented as external scaffolding, such as simple planning templates, basic tool-use wrappers, or short-horizon context management. This motivates a "built to delete" design principle: these harness components should provide explicit support when needed, while remaining modular enough to be simplified, retired, or replaced as models internalize the corresponding capabilities. By contrast, Layer 4 (Constraints & Guardrails) is less likely to be absorbed by model improvements alone, because its necessity depends on the autonomy, authority, and exposure granted to agents at deployment time. As more capable models enable higher-autonomy applications, they often receive broader action spaces, stronger privileges, and more consequential interfaces to external systems, expanding the operational risk and attack surface. Constraints and guardrails are therefore likely to remain a persistent and increasingly central part of production harnesses.

Disentangled and Cross-Harness Evaluation. A research priority is to separate model capability from harness capability in agent evaluation. Prevailing benchmarks often evaluate an agent as a monolithic system, obscuring whether success should be attributed to the underlying model, the harness, or their interface contract. The same model can behave differently under different tool schemas, memory policies, retry logic, state cleanup procedures, and environment-management mechanisms (Lee et al., 2026). Future protocols should report not only end-task success, but also harness-level contributions and cross-harness robustness: whether a model retains task competence under alternative tool schemas, feedback formats, memory policies, control loops, and execution substrates. This makes model–harness overfitting a measurable failure mode rather than an implicit source of benchmark variance.

Agent-Native Execution and Governance Infrastructure. A second research priority is to develop infrastructure for long-running, inspectable, and governable agent execution. Agentic workloads generate non-deterministic trajectories that span prompts, retrieved context, tool calls, intermediate decisions, memory updates, and changing environment states. Future harnesses therefore need semantic trajectory management for reconstruction, comparison, debugging, and replay, together with durable execution primitives for pausing, serialization, recovery, and controlled forking across API failures, rate limits, network partitions, and failed reasoning branches. At the governance layer, sandboxes and access-control mechanisms must also become more intent-aware: authority should be transient, task-contingent, and scoped to specific sub-goals rather than attached only to persistent users or services. More broadly, infrastructure may need to shift from managing applications, requests, or users to managing agent instances, raising open problems in elastic isolation, per-agent lifecycle management, and the economics of providing "one agent, one runtime" without physically dedicating one full stack to every idle agent.

Appendix B provides an extended discussion of these directions. Future agent harnesses should be deletable where models internalize their functions, but enforceable where evaluation, execution, and governance require system-level guarantees.

Limitations

This survey has several scope boundaries. First, we focus on the *infrastructure* dimension of agentic systems: model-intrinsic capabilities such as reasoning, planning, and instruction following are discussed only insofar as they interact with harness design. Second, the Agent Harness landscape evolves rapidly, especially in production-oriented systems; our survey therefore reflects the state of the literature and publicly documented systems at the time of writing, rather than a fixed or final catalog of harness architectures. Finally, for commercial and closed-source systems, we rely only on publicly available documentation and technical reports, and therefore focus on documented architectural patterns rather than implementation-level details.

Ethical Considerations

This survey discusses Agent Harnesses, whose capabilities in planning, memory, tool use, and environment interaction may introduce ethical risks, including hallucinations, bias, privacy leakage, unsafe actions, and reduced human oversight. We emphasize the need for responsible agent design, robust evaluation, sandboxed execution, privacy-aware memory management, and appropriate human supervision before real-world deployment. AI assistants were used only for language polishing and refinement during the preparation of this manuscript.

References

Emre Can Acikgoz, Cheng Qian, Jonas Hübotter, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. 2026. Tool-r0: Self-evolving llm agents for tool-learning from zero data. *arXiv preprint arXiv:2602.21320*.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. 2025. Self-evolving multi-agent simulations for realistic clinical interactions. *arXiv preprint arXiv:2503.22678*.

Adam AlSayyad, Kelvin Yuxiang Huang, and Richik Pal. 2026. Agenttrace: A structured logging framework for agent system observability. In *LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems*.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. 2026. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*.

Anthropic. 2025. Claude Code. Accessed: 2026-05-23.

Anthropic. 2025a. Introducing agent skills. https://www.anthropic.com/news/skills. Product Announcement, Oct 2025.

Anthropic. 2025b. Subagents. https://docs.anthropic.com/en/docs/claude-code/subagents. Claude Code documentation page describing custom AI sub agents.

Anthropic. 2026. Sandbox-runtime: A lightweight sandboxing tool for enforcing filesystem and network restrictions on arbitrary processes at the OS level, without requiring a container. https://github.com/anthropic-experimental/sandbox-runtime. GitHub repository.

Christoph Bühler, Matteo Biagiola, Luca Di Grazia, and Guido Salvaneschi. 2025. Securing ai agent execution. *arXiv preprint arXiv:2510.21236*.

Nghi DQ Bui. 2026. Building ai coding agents for the terminal: Scaffolding, harness, context engineering, and lessons learned. *arXiv preprint arXiv:2603.05344*.

Jinkun Chen, Sher Badshah, Xuemin Yu, and Sijia Han. 2025a. Static sandboxes are inadequate: Modeling societal complexity requires open-ended co-evolution in llm-based multi-agent simulations. *arXiv preprint arXiv:2510.13982*.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, et al. 2026a. Skillcraft: Can llm agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, et al. 2026b. Cua-skill: Develop skills for computer using agent. *arXiv preprint arXiv:2601.21123*.

Tianyu Chen, Dongrui Liu, Xia Hu, Jingyi Yu, and Wenjie Wang. 2026c. A trajectory-based safety audit of clawdbot (openclaw). *arXiv preprint arXiv:2602.14364*.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. 2025b. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*.

Daixuan Cheng, Shaohan Huang, Yuxian Gu, Huatong Song, Guoxin Chen, Li Dong, Wayne Xin Zhao, Ji-Rong Wen, and Furu Wei. 2026. Llm-in-sandbox elicits general agentic intelligence. *arXiv preprint arXiv:2601.16206*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. 2026. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 59(2):1–59.

João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe Jin, Jamie Callan, João Magalhães, Bruno Martins, et al. 2025. Deepresearchgym: A free, transparent, and reproducible evaluation sandbox for deep research. *arXiv preprint arXiv:2505.19253*.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732.

Ivan Daunis. 2025. A declarative language for building and orchestrating llm-powered agent workflows. *arXiv preprint arXiv:2512.19769*.

Xinhao Deng, Yixiang Zhang, Jiaqing Wu, Jiaqi Bai, Sibo Yi, Zhuoheng Zou, Yue Xiao, Rennai Qiu, Jianan Ma, Jialuo Chen, et al. 2026. Taming openclaw: Security analysis and mitigation of autonomous llm agent threats. *arXiv preprint arXiv:2603.11619*.

Mahir Labib Dihan, Tanzima Hashem, Mohammed Eunus Ali, and Md Rizwan Parvez. 2025. Weboperator: Action-aware tree search for autonomous agents in web environment. *arXiv preprint arXiv:2512.12692*.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks VS Lakshmanan, Qingyun Wu, and Victor Rühle. 2025. Best-route: Adaptive llm routing with test-time optimal compute. *arXiv preprint arXiv:2506.22716*.

Elzo Brito dos Santos Filho. 2026. Esaa: Event sourcing for autonomous agents in llm-based software engineering. *arXiv preprint arXiv:2602.23193*.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025a. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv e-prints*, pages arXiv–2505.

Yiming Du, Baojun Wang, Yifan Xiang, Zhaowei Wang, Wenyu Huang, Boyang Xue, Bin Liang, Xingshan Zeng, Fei Mi, Haoli Bai, et al. 2025b. Memory-t1: Reinforcement learning for temporal reasoning in multi-session agents. *arXiv preprint arXiv:2512.20092*.

Hanxi Fang, Supawit Chockchowwat, Hari Sundaram, and Yongjoo Park. 2025a. Large-scale evaluation of notebook checkpointing with ai agents. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025b. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8970–8986.

Tongtong Feng, Xin Wang, Zekai Zhou, Ren Wang, Yuwei Zhan, Guangyao Li, Qing Li, and Wenwu Zhu. 2025. Evoagent: Self-evolving agent with continual world model for long-horizon tasks. *arXiv preprint arXiv:2502.05907*.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*.

Tianyi Fu, Brian Jauw, and Mohan Sridharan. 2025. Combining llm, non-monotonic logical reasoning, and human-in-the-loop feedback in an assistive ai agent. In *2025 34th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 322–329. IEEE.

Janika Deborah Gajo, Gerarld Paul Merales, Jerome Escarcha, Brenden Ashley Molina, Gian Nartea, Emmanuel G Maminta, Juan Carlos Roldan, and Rowel O Atienza. 2025. Sari sandbox: A virtual retail store environment for embodied ai agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2369–2378.

Yudong Gao, Zongjie Li, Zimo Ji, Pingchuan Ma, Shuai Wang, et al. 2026. Skillreducer: Optimizing llm agent skills for token efficiency. *arXiv preprint arXiv:2603.29919*.

Xinshuai Guo, Jiayi Kuang, Linyue Pan, Yinghui Li, Yangning Li, Hai-Tao Zheng, Ying Shen, Di Yin, and Xing Sun. 2026. Evoconfig: Self-evolving multi-agent systems for efficient autonomous environment configuration. *arXiv preprint arXiv:2601.16489*.

Xiyu Guo, Shan Wang, Chunfang Ji, Xuefeng Zhao, Wenhao Xi, Yaoyao Liu, Qinglan Li, Chao Deng, and Junlan Feng. 2025. Towards generalized routing: Model and agent orchestration for adaptive and efficient inference. *arXiv preprint arXiv:2509.07571*.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

Chaoyue He, Xin Zhou, Di Wang, Hong Xu, Wei Liu, and Chunyan Miao. 2026. Harness engineering for language agents: The harness layer as control,

agency, and runtime. *Preprints*. https://doi.org/10.20944/preprints202603.1756.v1.

Yufei He, Ruoyu Li, Alex Chen, Yue Liu, Yulin Chen, Yuan Sui, Cheng Chen, Yi Zhu, Luca Luo, Frank Yang, et al. 2025. Enabling self-improving agents to learn at test time with human-in-the-loop guidance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1625–1653.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025a. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025b. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.

Haoyu Huang, Jiaxin Bai, Shujie Liu, Yang Wei, Hong Ting Tsang, Yisen Gao, Zhongwei Xie, Yufei Li, and Yangqiu Song. 2026. Deeprefine: Agent-compiled knowledge refinement via reinforcement learning. *Preprint*, arXiv:2605.10488.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. 2025. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Xuan Ren, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. 2026. Agentic proposing: Enhancing large language model reasoning via compositional skill synthesis. *arXiv preprint arXiv:2602.03279*.

Cheng Jiayang, Dongyu Ru, Lin Qiu, Yiyang Li, Xuezhi Cao, Yangqiu Song, and Xunliang Cai. 2026. Amemgym: Interactive memory benchmarking for assistants in long-horizon conversations. *arXiv preprint arXiv:2603.01966*.

Yihong Jin, Ze Yang, Xinhe Xu, Yihan Zhang, and Shuyang Jie. 2025. Adaptive fault tolerance mechanisms of large language models in cloud computing environments. In *2025 6th International Conference on Computer Engineering and Application (ICCEA)*, pages 754–757. IEEE.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. 2025. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*.

Adam Kaufman, James Lucassen, Tyler Tracy, Cody Rushing, and Aryan Bhatt. 2025. Basharena: A control setting for highly privileged ai agents. *arXiv preprint arXiv:2512.15688*.

Doyoung Kim, Zhiwei Ren, Jie Hao, Zhongkai Sun, Lichao Wang, Xiyao Ma, Zack Ye, Xu Han, Jun Yin, Heng Ji, et al. 2026. Beyond perfect apis: A comprehensive evaluation of llm agents under real-world api complexity. *arXiv preprint arXiv:2601.00268*.

LangChain. 2026a. Durable Execution in LangGraph. https://docs.langchain.com/oss/python/langgraph/durable-execution. LangChain documentation page describing durable execution in LangGraph.

LangChain. 2026b. The two patterns by which agents connect sandboxes. https://blog.langchain.com/the-two-patterns-by-which-agents-connect-sandboxes/. LangChain Blog.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. 2026. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*.

Frank Li. 2026a. Openclaw prism: A zero-fork, defense-in-depth runtime security layer for tool-augmented llm agents. *arXiv preprint arXiv:2603.11853*.

Han Li, Letian Zhu, Bohan Zhang, Rili Feng, Jiaming Wang, Yue Pan, Earl T Barr, Federica Sarro, Zhaoyang Chu, and He Ye. 2026a. Contextbench: A benchmark for context retrieval in coding agents. *arXiv preprint arXiv:2602.05892*.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, and Chaowei Xiao. 2026b. Agentdyn: A dynamic open-ended benchmark for evaluating prompt injection attacks of real-world agent security system. *arXiv preprint arXiv:2602.03117*.

Hao Li, Yankai Yang, G Edward Suh, Ning Zhang, and Chaowei Xiao. 2026c. Reasalign: Reasoning enhanced safety alignment against prompt injection attack. *arXiv preprint arXiv:2601.10173*.

Tianhao Li, Chuangxin Chu, Yujia Zheng, Bohan Zhang, Neil Zhenqiang Gong, and Chaowei Xiao. 2026d. A2asecbench: A protocol-aware security benchmark for agent-to-agent multi-agent systems. In *The Fourteenth International Conference on Learning Representations*.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. 2026e. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*.

Xiaoxiao Li. 2026b. When single-agent with skills replace multi-agent systems and when they fail. *arXiv preprint arXiv:2601.04748*.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. 2025. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management. *arXiv preprint arXiv:2508.21433*.

George Ling, Shanshan Zhong, and Richard Huang. 2026. Agent skills: A data-driven analysis of claude skills for extending large language model functionality. *arXiv preprint arXiv:2602.08004*.

Hanbing Liu, Chunhao Tian, Nan An, Ziyuan Wang, Pinyan Lu, Changyuan Yu, and Qi Qi. 2026a. Budget-constrained agentic large language models: Intention-based planning for costly tool use. *arXiv preprint arXiv:2602.11541*.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026b. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Shanyv Liu, Xuyang Yuan, Tao Chen, Zijun Zhan, Zhu Han, Danyang Zheng, Weishan Zhang, and Shaohua Cao. 2026c. Caster: Breaking the cost-performance barrier in multi-agent orchestration via context-aware strategy for task efficient routing. *arXiv preprint arXiv:2601.19793*.

Songyang Liu, Chaozhuo Li, Chenxu Wang, Jinyu Hou, Zejian Chen, Litian Zhang, Zheng Liu, Qiwei Ye, Yiming Hei, Xi Zhang, et al. 2026d. Clawkeeper: Comprehensive safety protection for openclaw agents through skills, plugins, and watchers. *arXiv preprint arXiv:2603.24414*.

Xunzhuo Liu, Bowei He, Xue Liu, Andy Luo, Haichen Zhang, and Huamin Chen. 2026e. Adaptive vision-language model routing for computer use agents. *arXiv preprint arXiv:2603.12823*.

Yi Liu, Zhihao Chen, Yanjun Zhang, Gelei Deng, Yuekang Li, Jianting Ning, Ying Zhang, and Leo Yu Zhang. 2026f. Malicious agent skills in the wild: A large-scale security empirical study. *arXiv preprint arXiv:2602.06547*.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. 2026g. Agent skills in the wild: An empirical study of security vulnerabilities at scale. *arXiv preprint arXiv:2601.10338*.

Ryan Lopopolo. 2026. Harness engineering: leveraging Codex in an agent-first world. https://openai.com/index/harness-engineering/. OpenAI Engineering Blog.

Yedidel Louck, Ariel Stulman, and Amit Dvir. 2025. Improving google a2a protocol: Protecting sensitive data and mitigating unintended harms in multi-agent systems. *arXiv preprint arXiv:2505.12490*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. 2025a. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. 2025b. Scaling llm multi-turn rl with end-to-end summarization-based context management. *arXiv preprint arXiv:2510.06727*.

Mingfei Lu, Mengjia Wu, Feng Liu, Jiawei Xu, Weikai Li, Haoyang Wang, Zhengdong Hu, Ying Ding, Yizhou Sun, Jie Lu, et al. 2026a. Choosing how to remember: Adaptive memory structures for llm agents. *arXiv preprint arXiv:2602.14038*.

Yuxing Lu, Yucheng Hu, Xukai Zhao, and Jiuxin Cao. 2026b. Dytopo: Dynamic topology routing for multi-agent reasoning via semantic matching. *arXiv preprint arXiv:2602.06039*.

Qianqian Luo, Qingming Lin, Liuchang Xu, Sensen Wu, Ruichen Mao, Chao Wang, Hailin Feng, Bo Huang, and Zhenhong Du. 2026. Geojson agents: a multi-agent llm architecture for geospatial analysis—function calling vs. code generation. *Big Earth Data*, pages 1–55.

Siyuan Ma, Bo Gao, Xiaojun Jia, Simeng Qin, Tianlin Li, Ke Ma, Xiaoshuang Jia, Wenqi Ren, and Yang Liu. 2026a. Odar: Principled adaptive routing for llm reasoning via active inference. *arXiv preprint arXiv:2602.23681*.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. 2026b. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*.

Narek Maloyan and Dmitry Namiot. 2026. Breaking the protocol: Security analysis of the model context protocol specification and prompt injection vulnerabilities in tool-integrated llm agents. *arXiv preprint arXiv:2601.17549*.

Md Motaleb Hossen Manik and Ge Wang. 2026. Openclaw agents on moltbook: Risky instruction sharing and norm enforcement in an agent-only social network. *arXiv preprint arXiv:2602.02625*.

Luoxi Meng, Henry Feng, Ilia Shumailov, and Earlence Fernandes. 2025. cellmate: Sandboxing browser ai agents. *arXiv preprint arXiv:2512.12594*.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*.

Zahra Moslemi, Keerthi Koneru, Yen-Ting Lee, Sheethal Kumar, and Ramesh Radhakrishnan. 2026. Polaris: Typed planning and governed execution for agentic ai in back-office automation. *arXiv preprint arXiv:2601.11816*.

João Moura. 2025. Crewai: Framework for orchestrating role-playing autonomous ai agents. https://github.com/crewaiinc/crewai. GitHub repository.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*.

Sriraam Natarajan, Saurabh Mathur, Sahil Sidheekh, Wolfgang Stammer, and Kristian Kersting. 2025. Human-in-the-loop or ai-in-the-loop? automate or collaborate? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28594–28600.

Kaiwen Ning, Jiachi Chen, Jingwen Zhang, Wei Li, Zexu Wang, Yuming Feng, Weizhe Zhang, and Zibin Zheng. 2026. Defining and detecting the defects of large language model-based autonomous agents. *IEEE Transactions on Software Engineering*, 52(3):1074–1093.

OpenAI. 2024. Swarm: An educational framework exploring ergonomic, lightweight multi-agent orchestration. https://github.com/openai/swarm. GitHub repository.

OpenClaw Contributors. 2026. Openclaw — personal ai assistant. https://github.com/openclaw/openclaw.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. 2026. Natural-language agent harnesses. *arXiv preprint arXiv:2603.25723*.

Parallel Web Systems. 2025. What is an agent harness in the context of large-language models? https://parallel.ai/articles/what-is-an-agent-harness. Parallel Web Systems Article.

Yun Piao, Hongbo Min, Hang Su, Leilei Zhang, Lei Wang, Yue Yin, Xiao Wu, Zhejing Xu, Liwei Qu, Hang Li, et al. 2025. Agentbay: A hybrid interaction sandbox for seamless human-ai intervention in agentic systems. *arXiv preprint arXiv:2512.04367*.

Artem Pshenitsyn, Aleksandr Panov, and Alexey Skrynnik. 2026. Camar: Continuous actions multi-agent routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29651–29659.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Stefano Rando, Luca Romani, Alessio Sampieri, Luca Franco, John Yang, Yuta Kyuragi, Fabio Galasso, and Tatsunori Hashimoto. 2025. Longcodebench: Evaluating coding llms at 1m context windows. *arXiv preprint arXiv:2505.07897*.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Xiyu Ren, Zhaowei Wang, Yiming Du, Zhongwei Xie, Chi Liu, Xinlin Yang, Haoyue Feng, Wenjun Pan, Tianshi Zheng, Baixuan Xu, et al. 2026. Memlens: Benchmarking multimodal long-term memory in large vision-language models. *arXiv preprint arXiv:2605.14906*.

Alexander Roman and Jacob Roman. 2026. Orchestral ai: A framework for agent orchestration. *arXiv preprint arXiv:2601.02577*.

Philipp Schmid. 2026. The importance of Agent Harness in 2026. https://www.philschmid.de/agent-harness-2026. Philipp Schmid Blog.

David Schmotz, Sahar Abdelnabi, and Maksym Andriushchenko. 2025. Agent skills enable a new class of realistic and trivially simple prompt injections. *arXiv preprint arXiv:2510.26328*.

David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. 2026. Skill-inject: Measuring agent vulnerability to skill file attacks. *arXiv preprint arXiv:2602.20156*.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, et al. 2025. Your agent may misevolve: Emergent risks in self-evolving llm agents. *arXiv preprint arXiv:2509.26354*.

Reshabh K Sharma. 2026. Contextcov: Deriving and enforcing executable constraints from agent instruction files. *arXiv preprint arXiv:2603.00822*.

Yuling Shi, Yichun Qian, Hongyu Zhang, Beijun Shen, and Xiaodong Gu. 2025. Longcodezip: Compress long context for code language models. *arXiv preprint arXiv:2510.00446*.

Shuang Sun, Huatong Song, Lisheng Huang, Jinhao Jiang, Ran Le, Zhihao Lv, Zongchao Chen, Yiwen Hu, Wenyang Luo, Wayne Xin Zhao, et al. 2026. Swe-world: Building software engineering agents in docker-free environments. *arXiv preprint arXiv:2602.03419*.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. 2025a. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. 2025b. SEAgent: Self-evolving computer use agent with autonomous learning from experience. *arXiv preprint arXiv:2508.04700*.

Balachandra Devarangadi Sunil, Isheeta Sinha, Piyush Maheshwari, Shantanu Todmal, Shreyan Mallik, and Shuchi Mishra. 2026. Memory poisoning attack and defense on memory based llm-agents. *arXiv preprint arXiv:2601.05504*.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2026. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. 2026. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

Joan C Timoneda and Sebastián Vallejo Vera. 2025. Memory is all you need: Testing how model memory affects llm performance in annotation tasks. *arXiv preprint arXiv:2503.04874*.

Maximilian Puelma Touzel, Sneheel Sarangi, Gayatri Krishnakumar, Busra Tugce Gurbuz, Austin Welch, Zachary Yang, Andreea Musulan, Hao Yu, Ethan Kosak-Hine, Tom Gibbs, et al. 2025. Sandboxsocial: A sandbox for social media using multimodal ai agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 11100–11103.

Universal Tool Calling Protocol. 2026. Codemode: Plug-and-play library to enable agents to call MCP and UTCP tools via code execution. https://github.com/universal-tool-calling-protocol/code-mode. GitHub repository.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. 2025. Compass: Enhancing agent long-horizon reasoning with evolving context. *arXiv preprint arXiv:2510.08790*.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. 2025a. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*.

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. 2026a. Skillorchestra: Learning to route agents via skill transfer. *arXiv preprint arXiv:2602.19672*.

Jingbo Wang, Sendong Zhao, Jiatong Liu, Haochun Wang, Wanting Li, Bing Qin, and Ting Liu. 2026b. Orchestrating intelligence: Confidence-aware routing for efficient multi-agent collaboration across multi-scale models. *arXiv preprint arXiv:2601.04861*.

Jingbo Wang, Sendong Zhao, Haochun Wang, Yuzheng Fan, Lizhe Zhang, Yan Liu, and Ting Liu. 2025b. Optimal-agent-selection: State-aware routing framework for efficient multi-agent collaboration. *arXiv preprint arXiv:2511.02200*.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. 2025c. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*.

Qianli Wang, Boyang Ma, Minghui Xu, and Yue Zhang. 2026c. When skills lie: Hidden-comment injection in llm agents. *arXiv preprint arXiv:2602.10498*.

Xilong Wang, Yinuo Liu, Zhun Wang, Dawn Song, and Neil Gong. 2026d. Websentinel: Detecting and localizing prompt injection attacks for web agents. *arXiv preprint arXiv:2602.03792*.

Yuhang Wang, Yuling Shi, Mo Yang, Rongrui Zhang, Shilin He, Heng Lian, Yuting Chen, Siyu Ye, Kai Cai, and Xiaodong Gu. 2026e. Swe-pruner: Self-adaptive context pruning for coding agents. *arXiv preprint arXiv:2601.16746*.

Yuhang Wang, Feiming Xu, Zheng Lin, Guangyu He, Yuzhe Huang, Haichang Gao, Zhenxing Niu, Shiguo Lian, and Zhaoxiang Liu. 2026f. From assistant to double agent: Formalizing and benchmarking attacks on openclaw for personalized local ai agent. *arXiv preprint arXiv:2602.08412*.

Zhaowei Wang, Lishu Luo, Haodong Duan, Weiwei Liu, Sijin Wu, Ji Luo, Shen Yan, Shuai Peng, Sihang Yuan, Chaoyi Huang, et al. 2026g. Training long-context vision-language models effectively with generalization beyond 128k context. *arXiv preprint arXiv:2605.13831*.

Zhaowei Wang, Wenhao Yu, Xiyu Ren, Jipeng Zhang, Yu Zhao, Rohit Saxena, Liang Cheng, Ginny Wong, Simon See, Pasquale Minervini, et al. 2025d. Mmlongbench: Benchmarking long-context vision-language models effectively and thoroughly. *Advances in Neural Information Processing Systems*, 38.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. 2025e. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025f. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. 2025. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*.

Ke Xiao, Haoze Zhang, Runze Mao, Han Li, and Zhi X Chen. 2026. Towards llm-enabled autonomous combustion research: A literature-aware agent for self-corrective modeling workflows. *arXiv preprint arXiv:2601.01357*.

Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. 2025. Improving the efficiency of llm agent systems through trajectory reduction. *arXiv preprint arXiv:2509.23586*.

Roy Xie, Deepak Gopinath, David Qiu, Dong Lin, Haitian Sun, Saloni Potdar, and Bhuwan Dhingra. 2026. Over-searching in search-augmented large language models. *arXiv preprint arXiv:2601.05503*.

Renjun Xu and Yang Yan. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. 2025a. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. 2025b. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Shuo Yang, Soyeon Caren Han, Xueqi Ma, Yan Li, Mohammad Reza Ghasemi Madani, and Eduard Hovy. 2026a. Evotool: Self-evolving tool-use policy optimization in llm agents via blame-aware mutation and diversity-aware selection. *arXiv preprint arXiv:2603.04900*.

Xianglin Yang, Yufei He, Shuo Ji, Bryan Hooi, and Jin Song Dong. 2026b. Zombie agents: Persistent control of self-evolving llm agents via self-reinforcing injections. *arXiv preprint arXiv:2602.15654*.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. 2026c. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*.

Zonghao Ying, Xiao Yang, Siyang Wu, Yumeng Song, Yang Qu, Hainan Li, Tianlin Li, Jiakai Wang, Aishan Liu, and Xianglong Liu. 2026. Uncovering security threats and architecting defenses in autonomous agents: A case study of openclaw. *arXiv preprint arXiv:2603.12644*.

Justin Young. 2025. Effective harnesses for long-running agents. Anthropic Engineering. Engineering note, accessed 2026-05-23.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Danlong Yuan, Wei Wu, Zhengren Wang, Xueliang Zhao, Huishuai Zhang, and Dongyan Zhao. 2026a. Swe-minisandbox: Container-free reinforcement learning for building software engineering agents. *arXiv preprint arXiv:2602.11210*.

Dun Yuan, Fuyuan Lyu, Ye Yuan, Weixu Zhang, Bowei He, Jiayi Geng, Linfeng Du, Zipeng Sun, Yankai Chen, Changjiang Han, et al. 2026b. Beyond message passing: Toward semantically aligned agent communication. *arXiv preprint arXiv:2604.02369*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572.

Kamer Ali Yuksel, Thiago Castro Ferreira, Mohamed Al-Badrashiny, and Hassan Sawaf. 2025. A multi-ai agent system for autonomous optimization of agentic ai solutions via iterative refinement and llm-driven feedback loops. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 52–62.

Caiqi Zhang, Menglin Xia, Xuchao Zhang, Daniel Madrigal, Ankur Mallick, Samuel Kessler, Victor Ruehle, and Saravan Rajmohan. 2026a. Budget-aware agentic routing via boundary-guided training. *arXiv preprint arXiv:2602.21227*.

Genghan Zhang, Weixin Liang, Olivia Hsu, and Kunle Olukotun. 2025a. Adaptive self-improvement llm agentic system for ml library development. *arXiv preprint arXiv:2502.02534*.

Genghan Zhang, Shaowei Zhu, Anjiang Wei, Zhenyu Song, Allen Nie, Zhen Jia, Nandita Vijaykumar, Yida Wang, and Kunle Olukotun. 2025b. Accelopt: A self-improving llm agentic system for ai accelerator kernel optimization. *arXiv preprint arXiv:2511.15915*.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025c. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*.

Guibin Zhang, Haiyang Yu, Kaiming Yang, Bingli Wu, Fei Huang, Yongbin Li, and Shuicheng Yan. 2026b. Evoroute: Experience-driven self-routing llm agent systems. *arXiv preprint arXiv:2601.02695*.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026c. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*.

Haozhen Zhang, Haodong Yue, Tao Feng, Quanyu Long, Jianzhu Bao, Bowen Jin, Weizhi Zhang, Xiao Li, Jiaxuan You, Chengwei Qin, et al. 2026d. Learning query-aware budget-tier routing for runtime agent memory. *arXiv preprint arXiv:2602.06025*.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. 2025d. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. 2025e. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. 2026e. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29892–29900.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. 2026f. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2026a. Lifelong learning of large language model based agents: A roadmap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(5):5552–5571.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuan Zhu, Baohua Dong, and Hangcheng Zhu. 2026b. Skillrouter: Retrieve-and-rerank skill selection for llm agents at scale. *arXiv preprint arXiv:2603.22455*.

Yupeng Zheng, Zebin Xing, Qichao Zhang, Bu Jin, Pengfei Li, Yuhang Zheng, Zhongpu Xia, Yaran Chen, and Dongbin Zhao. 2026c. Planagent: A multi-modal large language agent for closed-loop vehicle motion planning. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–14. Early Access.

Yusheng Zheng, Yanpeng Hu, Tong Yu, and Andi Quinn. 2025. Agentsight: System-level observability for ai agents using ebpf. In *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems*, pages 110–115.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. 2025a. Agentfly: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025b. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

Jared Zhu, Minhao Hu, and Junde Wu. 2026a. Swe context bench: A benchmark for context learning in coding. *arXiv preprint arXiv:2602.08316*.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. 2026b. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*.

A Survey Scope and Selection Criteria

We include works from 2023–2026 (mainly 2025–2026) that explicitly contribute to at least one harness-level function: orchestration, context or trajectory management, interaction surfaces, execution environments, or external constraints. We include peer-reviewed papers, arXiv preprints, open-source systems, and technical documentation when they introduce reusable infrastructure or widely adopted design patterns. We exclude works focused solely on model-intrinsic reasoning, pretraining, or benchmark performance unless they expose a harness-level mechanism.

B Extended Future Directions

This appendix expands on two implications of the taxonomy developed in the main text and the asymmetric design pressure summarized in Section 7. First, evaluation must disentangle model contributions from harness contributions. Second, runtime support must evolve toward agent-native execution and governance infrastructure.

B.1 Cross-Harness Evaluation and Attribution

The attribution problem outlined in Section 7 is compounded by a reproducibility problem. Even within a fixed model–harness pair, transient network states, residual inter-run interference, non-deterministic tool behavior, and brittle string-matching protocols can produce false negatives or uncontrolled variance. As a result, leaderboard scores often become noisy joint measurements of the model, the Harness, their interface contract, and the surrounding execution environment, instead of isolable estimates of model capability.

This motivates cross-harness generalization as a first-class agent capability. A model may perform well when post-trained or prompted for a specific harness's tool schemas, interaction patterns, memory policies, or feedback signals, yet fail to transfer under different execution substrates. Model–harness overfitting may cause a model to exploit the conventions of one Harness rather than adapt robustly to heterogeneous tool interfaces, state-management policies, and control loops. Future protocols should therefore report task success under multiple Harnesses, sensitivity to interface and state perturbations, and harness-side contributions such as trajectory compression, context management, trace compaction, stale-state cleanup, and robustness to environmental instability.

B.2 Toward Next-Generation Agent Infrastructure

The asymmetry also suggests a broader transition in computing infrastructure. If some planning, tool-use, and context-management scaffolds are gradually absorbed by stronger models, the remaining harness responsibilities shift toward primitives that models cannot provide internally: durable execution, inspectable state, governed authority, and runtime isolation for autonomous, long-running agents. We identify four emerging pillars: semantic trajectory management, durable execution, intent-aware governance, and agent-native runtime substrates.

Semantic Trajectory Management. Traditional version control systems (like Git's diffing algorithms) record versioned artifacts, such as source-code diffs, but not the non-deterministic trajectories by which agents produce them. Future harnesses require mechanisms that capture artifact changes together with observable plans, prompts, retrieved context, tool calls, intermediate decisions, and environment snapshots. These records should support trajectory reconstruction, comparison, and debugging of failures arising from planning, retrieval, tool use, or environmental drift.

Durable Agent Execution. Agentic workloads challenge the stateless assumptions of serverless computing. As execution horizons extend from seconds to hours or days, agents need runtimes that preserve progress across API failures, rate limits, network partitions, and reasoning dead-ends. This requires explicit execution-state primitives for pausing, serialization, replay, recovery, and controlled forking of trajectories to explore alternative plans without discarding prior work.

From Manual Forking to Intent-Aware Least-Privilege Execution. Current state-of-the-art infrastructures provide session forking, containerization, and access-control mechanisms to limit state contamination and resource exposure. However, these mechanisms remain largely operator-driven and identity-centric: humans decide when to fork a session, which sandbox to instantiate, and what permissions an agent should receive. This model is increasingly misaligned with agentic workloads, where authority is transient, task-contingent, and often scoped to a fleeting sub-goal rather than a persistent user or service identity.

We argue that next-generation agent harnesses must internalize this governance. Given an agent's declared objective, intermediate plan, and anticipated tool interactions, the harness should estimate the likely scope and risk of the intended action before execution. Within this environment, permissions could be dynamically minted as short-lived, narrow-scoped capabilities associated with the declared task, expected tool use, and current execution context, and revoked upon completion or deviation. Such intent-aware isolation couples environmental partitioning with least-privilege access control, reducing state contamination, agentic over-privilege, and lateral movement without requiring continuous human intervention.

Agent-Native Runtime. Agentic workloads shift the unit of infrastructure from applications, users, or requests to agent instances. A single user may spawn many long-lived or intermittently active agents, each requiring an execution sandbox, persistent workspace, structured database, memory store, and narrowly scoped credentials. Future harnesses should therefore evolve from orchestration layers into infrastructure synthesizers that provision and manage these per-agent runtime substrates on demand. The challenge is to provide the semantics of "one agent, one runtime" while retaining the economics of shared serverless systems: most agent-created resources remain idle, whereas a small fraction may experience sudden traffic spikes. This raises open problems in safe agent-generated schema evolution, elastic isolation without one physical instance per agent, and unified lifecycle management across databases, filesystems, memories, credentials, and observability traces.

Together, these problems suggest that the durable part of the agent harness will increasingly be less about compensating for weak model capabilities and more about providing the execution, isolation, and governance guarantees that cannot be internalized by a model alone.

C Summary Tables for the Agent Harness

Tables 1–6 provide a structured overview of all research works surveyed in this paper, organized by the four-layer taxonomy introduced in Section 2. Each entry is annotated with two binary indicators: whether it primarily introduces a *method or framework* (**Meth./Fw.**), and whether it includes a *benchmark or evaluation protocol* (**Bench./Eval.**). Entries where both indicators are negative (✗/✗) represent survey, roadmap, or analysis works included for their contextual or taxonomic relevance. A ✓ indicates that the work makes a primary contribution of that type, not merely that the paper discusses the topic.

Research Works	Meth. Fw.	Bench. Eval.	Featured Functionality	Layer Coverage			
				L1	*L2*	*L3*	*L4*
Young (Young, 2025)	✗	✗	long-running agent harness management	✓	✓	✗	✗
Pan et al. (Pan et al., 2026)	✓	✓	natural-language harness design	✓	✓	✓	✗
He et al. (He et al., 2026)	✓	✗	harness as control, agency, and runtime layer	✓	✓	✓	✓
Bui (Bui, 2026)	✓	✗	terminal coding agent scaffolding, context engineering, lessons learned	✓	✓	✓	✗
Lopopolo† (Lopopolo, 2026)	✗	✗	Codex-based harness engineering	✓	✓	✓	✗
Schmid† (Schmid, 2026)	✗	✗	agent harness importance analysis	✓	✓	✓	✓
Parallel† (Parallel Web Systems, 2025)	✗	✗	agent harness concept overview	✓	✓	✓	✓

Table 1: Agent Harness foundations: cross-layer conceptual works. †Blog/documentation.

Research Works	Domain	Meth. Fw.	Bench. Eval.	Featured Functionality	Autonomy Tier
Model & Agent Routing					
EvoRoute (Zhang et al., 2026b)	General	✓	✗	experience-driven self-routing	T2
Best-Route (Ding et al., 2025)	General	✓	✗	test-time optimal compute routing	T1
MASRouter (Yue et al., 2025)	General	✓	✗	multi-agent system routing learning	T2
Liu et al. (Liu et al., 2026e)	GUI	✓	✗	adaptive VLM routing for computer use	T3
CAMAR (Pshenitsyn et al., 2026)	General	✓	✗	continuous-action multi-agent routing	T2
SkillOrchestra (Wang et al., 2026a)	General	✓	✗	skill-transfer-based agent routing	T2
DyTopo (Lu et al., 2026b)	General	✓	✗	semantic-matching topology routing	T2
Wang et al. (Wang et al., 2026b)	General	✓	✗	confidence-aware multi-scale routing	T2
Zhang et al. (Zhang et al., 2026d)	General	✓	✗	query-aware budget-tier memory routing	T1
CASTER (Liu et al., 2026c)	General	✓	✗	context-aware task-efficient routing	T1
Zhang et al. (Zhang et al., 2026a)	General	✓	✗	boundary-guided budget-aware routing	T1
ODAR (Ma et al., 2026a)	General	✓	✗	active-inference adaptive routing	T2
Wang et al. (Wang et al., 2025b)	General	✓	✗	state-aware optimal agent selection	T2
Guo et al. (Guo et al., 2025)	General	✓	✗	generalized model-agent orchestration	T2
Multi-Agent Composition & Orchestration					
Compass (Wan et al., 2025)	General	✓	✗	evolving context for long-horizon reasoning	T3
Claude Subagents† (Anthropic, 2025b)	Code	✓	✗	custom AI subagent spawning	T3
Kimi K2.5 (Team et al., 2026)	GUI	✓	✗	visual agentic intelligence	T3
Swarm⋆ (OpenAI, 2024)	General	✓	✗	lightweight multi-agent orchestration	T2
CrewAI⋆ (Moura, 2025)	General	✓	✗	role-playing agent orchestration	T3
Daunis (Daunis, 2025)	General	✓	✗	declarative agent workflow language	T2
Orchestral AI (Roman and Roman, 2026)	General	✓	✗	general-purpose agent orchestration	T3
Autonomous Loop, Resilience & Human-in-the-Loop					
Jin et al. (Jin et al., 2025)	General	✓	✗	adaptive fault tolerance in cloud LLMs	T4
ESAA (dos Santos Filho, 2026)	General	✓	✗	event sourcing for autonomous agents	T4
Natarajan et al. (Natarajan et al., 2025)	General	✗	✗	human-in-the-loop vs AI-in-the-loop	T3
Fu et al. (Fu et al., 2025)	General	✓	✗	LLM + non-monotonic reasoning + HITL	T3
He et al. (He et al., 2025)	General	✓	✗	test-time learning with human guidance	T3
PlanAgent (Zheng et al., 2026c)	Embodied	✓	✗	closed-loop vehicle motion planning	T3
Yuksel et al. (Yuksel et al., 2025)	General	✓	✗	iterative refinement via LLM feedback	T3
Xiao et al. (Xiao et al., 2026)	Research	✓	✗	autonomous combustion research agent	T4
Ferrag et al. (Ferrag et al., 2025)	General	✗	✗	LLM reasoning to autonomous agents survey	T4

Table 2: Summary of Layer 1 (Execution & Orchestration) research works. Autonomy tiers describe the role left to the human operator: T1 requires direct operation, T2 supports collaborative control, T3 requires human approval at key steps, and T4 permits mostly autonomous execution with human observation. ⋆Open-source repository. †Blog/documentation.

Research Works	Domain	Meth. Fw.	Bench. Eval.	Capability			Featured Functionality
				Mem	*Cmp*	*Trj*	
Memory Systems							
Mem0 (Chhikara et al., 2025)	General	✓	✗	✓	✗	✗	scalable production long-term memory
Zep (Rasmussen et al., 2025)	General	✓	✗	✓	✗	✗	temporal knowledge graph memory
Timoneda et al. (Timoneda and Vera, 2025)	General	✓	✗	✓	✗	✗	memory effects on LLM annotation
AMemGym (Jiayang et al., 2026)	General	✗	✓	✓	✗	✗	interactive memory benchmarking
Hu et al. (Hu et al., 2025b)	General	✗	✓	✓	✗	✗	incremental multi-turn memory eval
Nemori (Nan et al., 2025)	General	✓	✗	✓	✗	✗	self-organizing cognitive memory
MemGPT (Packer et al., 2023)	General	✓	✗	✓	✓	✗	LLM as operating system with memory tiers
A-Mem (Xu et al., 2025)	General	✓	✗	✓	✗	✗	agentic self-organizing memory
MemAgent (Yu et al., 2025)	General	✓	✗	✓	✗	✗	multi-conv RL-based memory agent
G-Memory (Zhang et al., 2025c)	Multi	✓	✗	✓	✗	✓	hierarchical multi-agent memory tracing
HippoRAG (Gutiérrez et al., 2024)	General	✓	✗	✓	✗	✗	neurobiological long-term memory
SimpleMem (Liu et al., 2026b)	General	✓	✗	✓	✗	✗	efficient lifelong memory for agents
Yan et al. (Yan et al., 2025a)	Research	✓	✗	✓	✗	✗	agentic memory via deep research
Lu et al. (Lu et al., 2026a)	General	✓	✗	✓	✗	✗	adaptive memory structure selection
Zhu et al. (Zhu et al., 2026b)	General	✓	✗	✓	✓	✗	provenance-aware tiered memory
Zheng et al. (Zheng et al., 2026a)	General	✗	✗	✓	✗	✗	lifelong learning roadmap for agents
Sunil et al. (Sunil et al., 2026)	Security	✓	✗	✓	✗	✗	memory poisoning attack and defense
Context Compression							
ACON (Kang et al., 2025)	General	✓	✗	✗	✓	✗	long-horizon agent context compression
LongLLMLingua (Jiang et al., 2024)	General	✓	✗	✗	✓	✗	prompt compression via token scoring
Lu et al. (Lu et al., 2025b)	General	✓	✗	✓	✓	✗	summarization-based context management
LongCodeBench (Rando et al., 2025)	Code	✗	✓	✗	✓	✗	1M-token coding evaluation
Context-Folding (Sun et al., 2025a)	General	✓	✗	✗	✓	✓	hierarchical trajectory folding
Dai et al. (Dai et al., 2025)	General	✓	✗	✗	✓	✗	embedding-based context compressor
SWE-Pruner (Wang et al., 2026e)	Code	✓	✗	✗	✓	✗	self-adaptive context pruning for code
Lindenbauer et al. (Lindenbauer et al., 2025)	General	✓	✗	✗	✓	✗	observation masking vs summarization
LongCodeZip (Shi et al., 2025)	Code	✓	✗	✗	✓	✗	long-context code compression
Mem1 (Zhou et al., 2025b)	General	✓	✗	✓	✓	✗	synergize memory and reasoning
ContextBench (Li et al., 2026a)	General	✗	✓	✗	✓	✗	context retrieval benchmarking
MemRL (Zhang et al., 2026f)	General	✓	✗	✓	✓	✗	runtime RL on episodic memory
HiAgent (Hu et al., 2025a)	General	✓	✗	✓	✓	✗	hierarchical working memory management
SWE Context Bench (Zhu et al., 2026a)	Code	✗	✓	✗	✓	✗	context learning benchmarking
SafeSieve (Zhang et al., 2026e)	Multi	✓	✗	✗	✓	✗	progressive multi-agent comm pruning
Trajectory Persistence & Observability							
Xiao et al. (Xiao et al., 2025)	General	✓	✗	✗	✓	✓	trajectory reduction for efficiency
AgentTrace (AlSayyad et al., 2026)	General	✓	✗	✗	✗	✓	structured logging for observability
AgentSight (Zheng et al., 2025)	General	✓	✗	✗	✗	✓	eBPF-based system-level observability
LangGraph Durable† (LangChain, 2026a)	General	✓	✗	✗	✗	✓	fault-tolerant durable execution
Fang et al. (Fang et al., 2025a)	Code	✗	✓	✗	✗	✓	notebook checkpointing evaluation

Table 3: Summary of Layer 2 (Context & Trajectory Management) research works—Part I: Memory (Mem), Compression (Cmp), and Trajectory Persistence & Observability (Trj). †Blog/documentation.

Research Works	Domain	Meth. Fw.	Bench. Eval.	Capability			Featured Functionality
				Evo	*Skl*	*Sec*	
Self-Evolving Architectures							
Darwin Godel Machine (Zhang et al., 2025d)	General	✓	✗	✓	✗	✗	open-ended self-improving evolution
Shao et al. (Shao et al., 2025)	General	✗	✗	✓	✗	✓	emergent risks in self-evolution
Live-SWE-Agent (Xia et al., 2025)	Code	✓	✗	✓	✗	✗	on-the-fly SWE agent self-evolution
Zhang et al. (Zhang et al., 2025e)	General	✓	✗	✓	✗	✗	evolving contexts for self-improvement
GEPA (Agrawal et al., 2025)	General	✓	✗	✓	✗	✗	reflective prompt evolution
Dynamic Cheatsheet (Suzgun et al., 2026)	General	✓	✗	✓	✗	✗	test-time learning with adaptive memory
Zhang et al. (Zhang et al., 2025a)	Code	✓	✗	✓	✗	✗	self-improvement for ML library dev
AccelOpt (Zhang et al., 2025b)	Code	✓	✗	✓	✗	✗	self-improving kernel optimization
AgentFly (Zhou et al., 2025a)	General	✓	✗	✓	✗	✗	fine-tuning agents without LLM FT
Chen et al. (Chen et al., 2025b)	General	✓	✗	✓	✗	✗	LLM self-improve through co-evolution
RAGEN (Wang et al., 2025e)	General	✓	✗	✓	✗	✗	self-evolution via multi-turn RL
Evo-Memory (Wei et al., 2025)	General	✗	✓	✓	✗	✗	benchmarking self-evolving memory
WebEvolver (Fang et al., 2025b)	Web	✓	✗	✓	✗	✗	co-evolving web world model
SEAgent (Sun et al., 2025b)	GUI	✓	✗	✓	✓	✗	self-evolving computer use agent
Almansoori et al. (Almansoori et al., 2025)	Research	✓	✓	✓	✗	✗	self-evolving clinical simulations
EvoAgent (Feng et al., 2025)	Embodied	✓	✗	✓	✗	✗	continual world model for long-horizon
Tool-R0 (Acikgoz et al., 2026)	General	✓	✗	✓	✓	✗	self-evolving tool learning from zero
EvoTool (Yang et al., 2026a)	General	✓	✗	✓	✓	✗	blame-aware tool-use optimization
AutoSkill (Yang et al., 2026c)	General	✓	✗	✓	✓	✗	experience-driven lifelong skill evolution
MemSkill (Zhang et al., 2026c)	General	✓	✗	✓	✓	✗	learning and evolving memory skills
EvoConfig (Guo et al., 2026)	Multi	✓	✗	✓	✗	✗	self-evolving multi-agent configuration
Xie et al. (Xie et al., 2026)	Web	✗	✓	✗	✗	✗	over-searching in search-augmented LLMs
Agentic Skills							
Wang et al. (Wang et al., 2025f)	General	✓	✗	✗	✓	✗	inducing programmatic skills
SkillCraft (Chen et al., 2026a)	General	✗	✓	✗	✓	✗	tool-use skill learning evaluation
SoK: Skills (Jiang et al., 2026)	General	✗	✗	✓	✓	✓	systematization of agentic skills
SkillReducer (Gao et al., 2026)	General	✓	✗	✗	✓	✗	token-efficient skill optimization
Xu et al. (Xu and Yan, 2026)	General	✗	✗	✓	✓	✓	skill architecture, acquisition, security
Ling et al. (Ling et al., 2026)	Code	✗	✗	✗	✓	✓	data-driven Claude skill analysis
SkillRouter (Zheng et al., 2026b)	General	✓	✗	✗	✓	✗	retrieve-and-rerank skill selection
Li (Li, 2026b)	General	✓	✗	✗	✓	✗	single-agent skills vs multi-agent
EvoSkill (Alzubi et al., 2026)	Multi	✓	✗	✓	✓	✗	automated multi-agent skill discovery
SkillsBench (Li et al., 2026e)	General	✗	✓	✗	✓	✗	skill benchmarking across diverse tasks
CUA-Skill (Chen et al., 2026b)	GUI	✓	✗	✗	✓	✗	skills for computer-using agents
Claude Skills† (Anthropic, 2025a)	Code	✓	✗	✗	✓	✗	agent skill platform launch
SAGE (Wang et al., 2025c)	General	✓	✗	✓	✓	✗	RL-based self-improving skill library
Jiao et al. (Jiao et al., 2026)	General	✓	✗	✗	✓	✗	compositional skill synthesis
SkillClaw (Ma et al., 2026b)	Multi	✓	✗	✓	✓	✗	collective skill evolution via cloud sharing
Skills Security							
Schmotz et al. (Schmotz et al., 2025)	Security	✗	✗	✗	✓	✓	skill-based prompt injection analysis
Liu et al. (Liu et al., 2026g)	Security	✗	✗	✗	✓	✓	skill security vulnerabilities at scale
Liu et al. (Liu et al., 2026f)	Security	✗	✗	✗	✓	✓	malicious skill detection study
Wang et al. (Wang et al., 2026c)	Security	✗	✗	✗	✓	✓	hidden-comment skill injection
Yang et al. (Yang et al., 2026b)	Security	✓	✗	✓	✓	✓	persistent control via self-reinforcing injection

Table 4: Summary of Layer 2 (Context & Trajectory Management) research works—Part II: Self-Evolving (Evo), Agentic Skills (Skl), and Skills Security (Sec). †Blog/documentation.

Research Works	Domain	Meth. Fw.	Bench. Eval.	Interaction Modality	Featured Functionality
Standardized Protocols & Interaction Surface					
Chowa et al. (Chowa et al., 2026)	General	✗	✗	API	LLM as autonomous agent review
Ning et al. (Ning et al., 2026)	General	✓	✗	API	LLM agent defect detection
Wolflein et al. (Wölflein et al., 2025)	General	✓	✗	API	agents making agent tools
UTCP Code-Mode⋆ (Universal Tool Calling Protocol, 2026)	Code	✓	✗	Protocol	MCP/UTCP via code execution
Yuan et al. (Yuan et al., 2026b)	Multi	✓	✗	A2A	semantically aligned agent communication
Louck et al. (Louck et al., 2025)	Security	✓	✗	A2A	A2A protocol sensitive data protection
A2ASecBench (Li et al., 2026d)	Security	✗	✓	A2A	protocol-aware A2A security benchmark
UI-TARS (Qin et al., 2025)	GUI	✓	✗	GUI	native automated GUI interaction
GeoJSON Agents (Luo et al., 2026)	General	✓	✗	API/Code	function calling vs code generation
Kim et al. (Kim et al., 2026)	General	✗	✓	API	real-world API complexity evaluation
Tool Use & Code Execution					
WebOperator (Dihan et al., 2025)	Web	✓	✗	GUI	action-aware web tree search
Terminal-Bench (Merrill et al., 2026)	Code	✗	✓	CLI	CLI task benchmarking
Liu et al. (Liu et al., 2026a)	General	✓	✗	API	budget-constrained tool planning
ToolSandbox (Lu et al., 2025a)	General	✗	✓	API	stateful tool-use evaluation

Table 5: Summary of Layer 3 (Interaction Surface & Execution Environment) research works. Modalities: API (function calling), CLI (terminal), GUI (visual), Code (code execution), Protocol (MCP/UTCP), A2A (agent-to-agent). ⋆Open-source repository.

Research Works	Domain	Meth. Fw.	Bench. Eval.	Enforcement			Featured Functionality
				Pre	*Run*	*Post*	
Sandboxing & Execution Environments							
LangChain† (LangChain, 2026b)	General	✗	✗	✗	✓	✗	agent-sandbox connection patterns
SWE-MiniSandbox (Yuan et al., 2026a)	Code	✓	✗	✗	✓	✗	container-free RL sandbox
UI-TARS-2 (Wang et al., 2025a)	GUI	✓	✗	✗	✓	✗	multi-turn RL GUI agent sandbox
LLM-in-Sandbox (Cheng et al., 2026)	General	✓	✗	✗	✓	✗	sandbox for agentic intelligence
Sari Sandbox (Gajo et al., 2025)	Web	✗	✓	✗	✓	✗	virtual retail store environment
SandboxSocial (Touzel et al., 2025)	Web	✗	✓	✗	✓	✗	social media simulation sandbox
Cellmate (Meng et al., 2025)	Web	✓	✗	✗	✓	✗	browser AI agent sandboxing
AgentBay (Piao et al., 2025)	General	✓	✗	✗	✓	✗	hybrid human-AI interaction sandbox
Chen et al. (Chen et al., 2025a)	General	✗	✗	✗	✓	✗	static sandboxes are inadequate
DeepResearchGym (Coelho et al., 2025)	Research	✗	✓	✗	✓	✓	reproducible deep research evaluation
SWE-World (Sun et al., 2026)	Code	✓	✗	✗	✓	✗	Docker-free SWE environments
R2E-Gym (Jain et al., 2025)	Code	✗	✓	✗	✓	✓	procedural environments with hybrid verifiers
Governance Boundaries							
POLARIS (Moslemi et al., 2026)	General	✓	✗	✓	✓	✗	typed planning and governed execution
ContextCov (Sharma, 2026)	General	✓	✗	✓	✗	✓	executable constraint enforcement
Sandbox-Runtime⋆ (Anthropic, 2026)	Code	✓	✗	✓	✓	✗	OS-level filesystem/network sandboxing
Buhler et al. (Bühler et al., 2025)	General	✓	✗	✓	✓	✓	agent execution security analysis
BashArena (Kaufman et al., 2025)	Code	✗	✓	✗	✓	✓	highly-privileged agent control setting
Maloyan et al. (Maloyan and Namiot, 2026)	Security	✗	✗	✓	✗	✗	MCP specification security analysis
Agent Injection & Defense							
Skill-Inject (Schmotz et al., 2026)	Security	✗	✓	✓	✗	✗	skill file attack measurement
AgentDyn (Li et al., 2026b)	Security	✗	✓	✓	✓	✗	dynamic prompt injection benchmark
WebSentinel (Wang et al., 2026d)	Web	✓	✗	✗	✓	✗	web agent injection detection
ReasAlign (Li et al., 2026c)	General	✓	✗	✗	✓	✗	reasoning-enhanced injection safety
Wang et al. (Wang et al., 2026f)	Security	✗	✓	✓	✗	✗	formalizing attacks on OpenClaw
Deng et al. (Deng et al., 2026)	Security	✗	✗	✓	✓	✗	OpenClaw security analysis and mitigation
Ying et al. (Ying et al., 2026)	Security	✗	✗	✓	✗	✓	OpenClaw threat architecture and defenses
ClawKeeper (Liu et al., 2026d)	Security	✓	✗	✓	✓	✓	comprehensive safety via skills and watchers
Chen et al. (Chen et al., 2026c)	General	✗	✓	✗	✗	✓	trajectory-based safety audit
OpenClaw PRISM (Li, 2026a)	Security	✓	✗	✓	✓	✓	zero-fork defense-in-depth runtime
Manik et al. (Manik and Wang, 2026)	Security	✗	✗	✗	✓	✗	risky instruction sharing and norm enforcement

Table 6: Summary of Layer 4 (Constraints & Guardrails) research works. Enforcement timing: Pre (pre-execution validation), Run (runtime isolation), Post (post-execution audit). ⋆Open-source repository. †Blog/documentation.