AI Box

Agent Systems with Harness Engineering

Xinyu Tang[1,4*], **Han Peng**[1,4*], **Guoxin Chen**[1,4], **Yuze Shi**[2], **Zitao Su**[1,4], **Peiyu Liu**[3], **Wayne Xin Zhao**[1,4†], **Yawen Li**[2†], **Zhe Xue**[2]
[1]Gaoling School of Artificial Intelligence, Renmin University of China,
[2]Beijing University of Posts and Telecommunications,
[3]University of International Business and Economic,
[4]Beijing Key Laboratory of Research on Large Models and Intelligent Governance.
txy20010310@163.com, panospeng@ruc.edu.cn, batmanfly@gmail.com

Abstract

Developing effective agent systems has been a long-standing frontier in artificial intelligence. The recent emergence of LLM-based agents marks a paradigm shift, yet even highly capable models deployed without adequate system support exhibit systematic failures in long-horizon settings, including loss of intermediate goals, incorrect tool invocation, and inability to incorporate environmental feedback. These failures reflect not gaps in model training but a structural mismatch between the single-turn generative interface of LLMs and the stateful, iterative nature of real-world problem solving. In this paper, we formalize the surrounding infrastructure as the *harness* and introduce *harness engineering* as the joint optimization of both the harness and the model it supports. The relationship between the two is inherently *synergistic*: a well-designed harness unlocks latent model potential through structured memory, error recovery, and adaptive context management, while a capable model enables the harness to implement sophisticated workflows that would be infeasible with a weaker reasoner. We present a structured taxonomy of harness components, spanning agent workflows, memory systems, skill libraries, and multi-agent orchestration, and analyze how each contributes to system-level performance, distilling design principles and architectural trade-offs from current practice. We further review optimization strategies from both the scaffold side and the model side, and summarize evaluation benchmarks across software engineering, deep research, tool use, computer use, and scientific discovery. By shifting focus from *what* agents can accomplish to *how* the surrounding infrastructure enables them to do so, this paper aims to serve as a practical reference for building reliable, scalable, and controllable agent systems through principled harness engineering.

Keywords: Harness Engineering, Agent Systems
Date: May 18, 2026.
Repository: https://github.com/RUCAIBox/awesome-agent-harness

[*]Equal contribution.
[†]Corresponding author.

Contents

1 Introduction **4**

2 Foundations of Harness Engineering **6**

2.1 The Concept of Harness Engineering 6

2.2 Evolution of Harness Engineering 8

2.2.1 Action Interface: Connecting Model to Environment 8

2.2.2 Workflow Infrastructure: Orchestrating Persistent Workspaces 8

2.2.3 User-Centric Persistence: Continuity Across Sessions and Channels 8

2.3 Connections with Related Concepts 9

3 The Design of the Harness **9**

3.1 Agent Workflow 9

3.1.1 Environment Perception 10

3.1.2 Task Planning 11

3.1.3 Action Execution 12

3.2 Memory Systems 12

3.2.1 Short-term Memory 13

3.2.2 Long-term Memory 13

3.3 Skill Libraries 14

3.3.1 Skill Acquisition 14

3.3.2 Skill Management 15

3.3.3 Skill Maintenance 15

3.4 Multi-agent Orchestration 16

3.4.1 Coordination Architectures 16

3.4.2 Communication Mechanisms 17

3.5 A Comparative View of Representative Agent Systems 18

4 Model Adaptation for Harness **19**

4.1 Context Engineering 20

4.1.1 Context Design 20

4.1.2 Context Management 20

4.2 Agentic Training 21

4.2.1 Environment Construction 21
4.2.2 Training Optimization Algorithms 22
4.2.3 Supervision Signals 23
4.2.4 Infrastructure 23

5 Harness Capacity and Evaluation **24**
5.1 Deep Research 24
5.2 Software Engineering 26
5.3 Tool Use and Function Calling 26
5.4 Computer Use and GUI Grounding 27
5.5 ML Engineering and Scientific Research 27

6 Future Directions **28**
6.1 Efficiency 28
6.2 Safety 29
6.3 Continual Learning 30
6.4 State and Environment Modeling 30
6.5 Embodied Harnesses 31
6.6 Evaluation 32

7 Conclusion **32**

1 Introduction

Developing effective agent systems has been a long-standing frontier in artificial intelligence [1], tracing back to early symbolic agents [2] and rule-based designs [3] that excelled in simplified, structured environments but struggled with complexity and scalability. Over the decades, progress through reinforcement learning [4] and robotic agents [5] brought adaptive behaviors; yet real-world deployment remained hindered by brittle decision-making and limited generalization.

The recent emergence of large language model (LLM)-based agent systems [6] has marked a paradigm shift: by leveraging rich commonsense knowledge, natural language interaction, and complex reasoning, LLMs empower agents to plan, use tools, and adapt in open-ended scenarios—dramatically enhancing their practicality for real-world applications such as software automation [7, 8], scientific discovery [9, 10], and personal assistance [11, 12]. Given these remarkable capabilities, initial LLM-based agent development placed significant emphasis on the models themselves rather than on the supporting mechanisms within the agent systems.

However, as tasks grow increasingly complicated, a fundamental challenge has emerged: even highly capable LLMs deployed without adequate system support exhibit systematic failures in long-horizon settings [13, 14]. For example, models frequently lose track of intermediate goals during multi-step planning, invoke tools with incorrect arguments or at inappropriate moments, and fail to incorporate environmental feedback to correct earlier mistakes [15]. These shortcomings are not merely gaps in model training that future scaling will close; they reflect a structural mismatch between the single-turn generative interface of LLMs and the stateful, iterative nature of real-world problem solving. A growing consensus has consequently emerged: the practical capability of an AI system depends as much on the design of its supporting harness as on the generative capacity of the underlying model.

Therefore, it is equally important to consider the design and development of supporting systems. In the context of LLMs, agent systems were initially defined as holistic frameworks comprising key modules—memory, tools, planning, and action—built atop LLMs. The field has since expanded to encompass prompt and context engineering [16, 17], which improves LLM capacities without altering model weights. More recently, agentic frameworks have advanced by redesigning the agent–environment interface itself. Unlike earlier systems relying on free-form text actions or limited tool calls, modern frameworks such as OpenClaw [11] and Hermes Agent [12] introduce highly capable action spaces (*e.g.,* bash commands, file edits, search operations), stateful sandboxed execution environments, and direct computer control—aiming to enable autonomous, long-horizon task execution without human oversight. These design choices shift the bottleneck from raw LLM reasoning to robust system scaffolding, transforming agents from fragile proof-of-concepts into deployable systems for real-world tasks.

This recognition—that practical capability depends as much on system scaffolding as on the model itself—naturally invites a more precise characterization of what this "supporting system" entails. In recent literature, the term harness has been adopted to refer specifically to the infrastructure that surrounds and extends a foundation model [13, 14]. We accordingly decompose an agent system into two fundamental components: a foundation model that serves as the reasoning engine responsible for language understanding, generation, and decision-making, and a supporting harness that governs execution flow, tool invocation, memory management, context construction, safety enforcement, and multi-step coordination with external environments [18, 19, 20, 21]. Without a harness, a model remains a passive text generator; only when embedded within a properly engineered harness does it become a functional agent.

Crucially, the relationship between these two components is not merely compositional but inherently *synergistic*: improvements to one component unlock latent potential in the other, producing system-level gains that neither could achieve in isolation. A more capable model, for instance, can follow more nuanced instructions, enabling the harness to implement sophisticated multi-step workflows that would be infeasible with a weaker reasoner. Conversely, a well-designed harness that provides structured memory retrieval, error recovery mechanisms, and adaptive context management allows even a moderately capable model to sustain coherent performance over long-horizon tasks that would otherwise exceed its standalone capacity. Harness engineering advances this synergy along two complementary fronts. On the scaffold side, it designs more effective workflows, memory architectures, skill libraries, and orchestration mechanisms that structure and channel model behavior. On the model side, it adapts the foundation model itself through context engineering and agentic training to

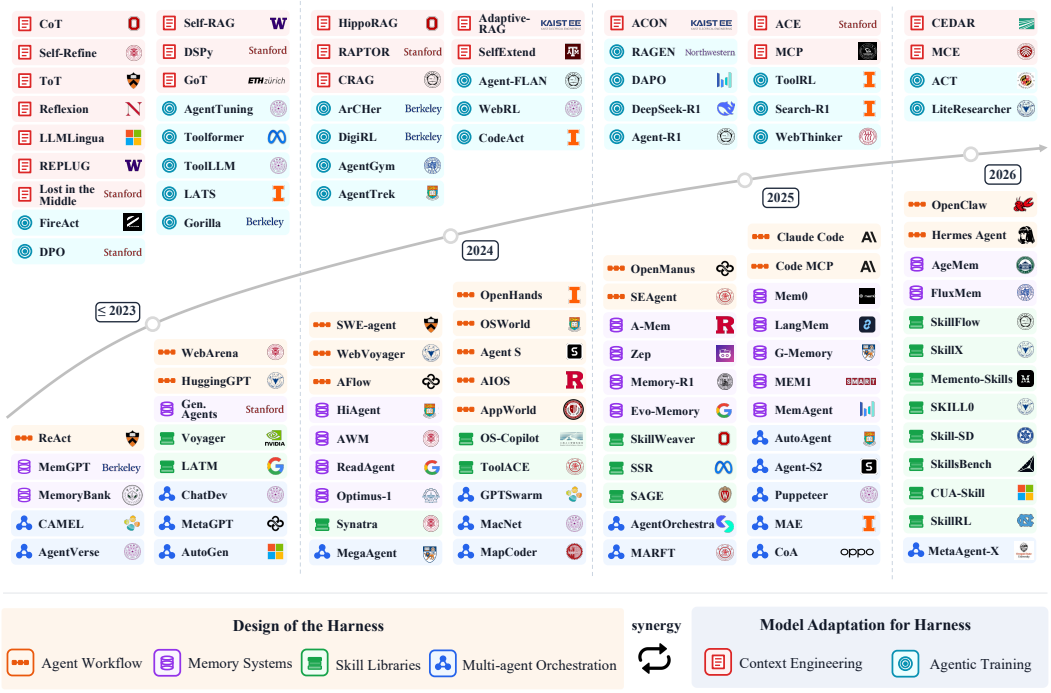

Figure 1: Timeline of representative agent systems with harness engineering. The lower portion presents scaffold-side optimization, encompassing four categories: agent workflow, memory systems, skill libraries, and multi-agent orchestration. The upper portion presents model-side adaptation, comprising two categories: context engineering and agentic training. These two dimensions are inherently synergistic—scaffold design shapes the interaction patterns that inform model adaptation, while improved model capabilities in turn enable more sophisticated scaffold architectures—together driving the development of more capable, robust, and generalizable agent systems.

more effectively exploit the capabilities afforded by the harness. We accordingly formalize *harness engineering* as the joint optimization of both components: designing the harness to accommodate the model's strengths and limitations, while simultaneously adapting the model to leverage the harness's affordances. It is precisely this bidirectional synergistic optimization that endows agent systems with the reliability, adaptability, and controllability required for complex real-world tasks.

In this paper, we focus on examining how the supporting infrastructure enables effective agent behavior, shifting the analytical focus from agent capabilities to the mechanisms that produce them. Specifically, the contributions of this paper are threefold:

- We formalize harness engineering as a distinct research paradigm, establishing its conceptual foundations and clarifying its relationship to adjacent paradigms. This provides a unified vocabulary for understanding the infrastructure that elevates language models into autonomous agents.
- We present a structured taxonomy of harness components, spanning agent workflows, memory systems, skill libraries, and multi-agent orchestration, and systematically analyze how each contributes to system-level performance, distilling design principles, architectural trade-offs, and empirical best practices from current practice.
- We comprehensively review optimization strategies that operate across both the scaffold and the underlying model, and summarize evaluation benchmarks that assess harness effectiveness in domains such as deep research, software engineering, tool use, computer use, and scientific discovery.

The remainder of this paper is organized as follows. Section 2 establishes the conceptual foundations of harness engineering. Section 3 details the principal components of harness design. Section 4 examines model adaptation techniques, including context engineering and agentic training. Section 5 summarizes evaluation benchmarks across deep research, software engineering, tool use, computer use, and scientific research. Section 6 identifies open challenges and future directions.

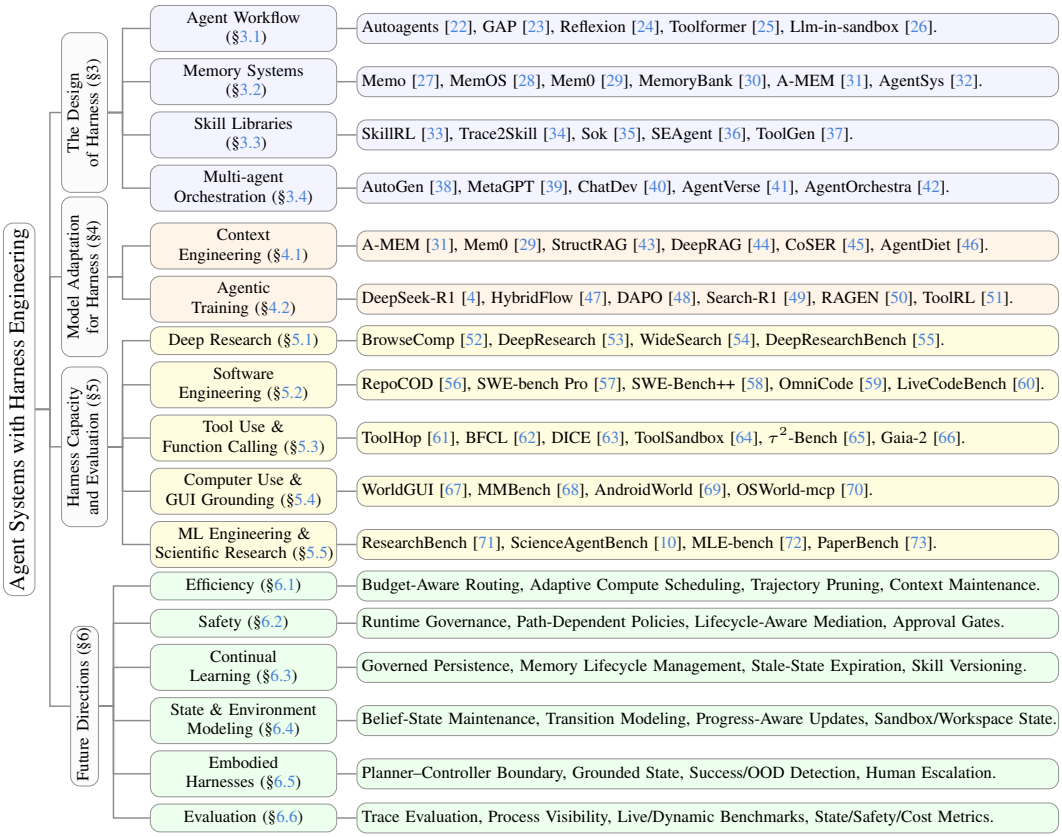

Figure 2: Overview of the structure of this paper.

2 Foundations of Harness Engineering

The practical capability of an agent system depends not only on the foundation model but equally on the orchestration layer that supports it [74, 75]. This layer structures reasoning steps, mediates tool interactions, and governs iterative execution across extended task horizons [76, 77]. We refer to its systematic study and design as *harness engineering*, with particular attention to how its scope expands as agentic systems evolve from simple model-environment coupling to increasingly capable and persistent forms of assistance. In this section, we introduce the concept through three progressively broader perspectives (Section 2.1), trace its evolution across three evolutionary phases (Section 2.2), and clarify its relationship to related concepts (Section 2.3).

2.1 The Concept of Harness Engineering

The notion of harness engineering admits multiple interpretations. We organize these into three progressively broader perspectives, each subsuming the previous one.

Perspective I: Harness Engineering as Structural Decomposition. The first and most basic perspective offers a structural decomposition: an agent system consists of two constitutive elements, a foundation model and its supporting harness. The foundation model provides core capabilities in language understanding, reasoning, and generation, while the harness constitutes the orchestration layer that mediates all interactions between the model and the external environment. Concretely, the harness governs tool invocation (*e.g.,* API calls and code execution), context maintenance (*e.g.,* conversation history and memory management), state persistence (*e.g.,* storage of intermediate results), and execution control (*e.g.,* termination conditions and retry policies). System-level performance is thus jointly determined by both components: a foundation model without a harness remains a reasoning engine capable of producing outputs yet unable to engage in sustained, goal-directed behavior; only when coupled with a properly designed harness does it become a fully functional

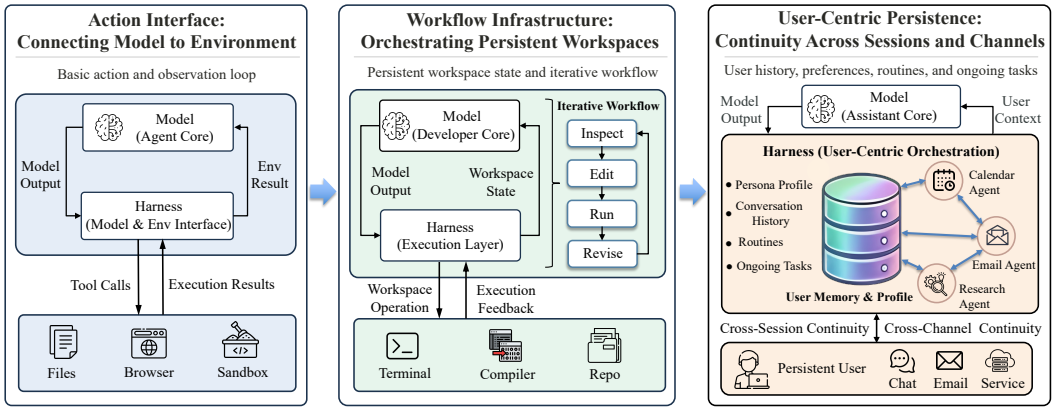

Figure 3: Three layers in the evolution of harness architecture, illustrating the progressive expansion of its scope and responsibility. **Layer 1 (Action Interface: Connecting Model to Environment):** The harness translates model outputs into executable tool calls (*e.g.,* file operations, browser interactions, sandbox execution) and returns environmental observations, establishing a basic action-observation loop. **Layer 2 (Workflow Infrastructure: Orchestrating Persistent Workspaces):** The harness maintains persistent workspace state and orchestrates iterative workflows of inspecting code, applying edits, running commands, and revising based on execution feedback within technical environments such as terminals, compilers, and version-control systems. **Layer 3 (User-Centric Persistence: Continuity Across Sessions and Channels):** The harness manages user profiles, memory, conversation history, routines, and ongoing tasks, coordinating multiple specialized agents and providing cross-session and cross-channel continuity.

agent that can execute tasks in real-world environments. This perspective, however, remains primarily descriptive, for it identifies what surrounds the model without prescribing how the supporting structure should be designed or improved.

Perspective II: Harness Engineering as Framework Construction. The second perspective moves from description to implementation by equating harness engineering with the design and construction of agent frameworks. From this viewpoint, building a harness extends well beyond wrapping a model with a thin interface layer; it entails creating a complete, reusable software infrastructure that supports multi-step interaction, tool integration, and complex agentic workflows. The objectives of harness engineering, under this interpretation, coincide with those of agent framework development: to furnish developers with extensible, robust, and ready-to-deploy infrastructure for constructing agentic applications. While this perspective captures much of the practical engineering effort, it focuses on a single direction of optimization and does not address the complementary question of how the model itself can be adapted to better exploit the supporting infrastructure.

Perspective III: Harness Engineering as Synergistic Enhancement. This paper adopts a broader and more integrated view. Rather than treating harness engineering as either structural decomposition (Perspective I) or framework construction (Perspective II), we position it as a synergistic optimization methodology for agent systems. Whereas Perspective I focuses on the question "what constitutes an agent system?" and Perspective II examines "how to build the scaffold around a model?", Perspective III explores "how can the scaffold and the model be jointly optimized to maximize system-level performance?" This synergistic formulation encompasses two complementary directions. The first direction targets optimizations of the scaffold supporting the model (Section 3), including the construction of agent workflows, the design of memory systems, the curation of skill libraries, and the orchestration of multi-agent collaboration. The second direction targets optimizations of the model itself to better cooperate with the harness (Section 4), encompassing context engineering and agentic training. Neither direction alone is sufficient: a well-architected scaffold cannot compensate for a model lacking fundamental reasoning capabilities, nor can a highly capable model realize its full potential within a poorly structured system. It is through the synergistic optimization of both sides that agent systems can attain the reliability, scalability, and controllability required for complex, long-horizon tasks in real-world settings.

2.2 Evolution of Harness Engineering

As agentic systems have been deployed in increasingly complex environments, the harness has had to support a broader range of actions, longer-horizon states, and more powerful forms of delegated control. To meet these growing demands, harness engineering has evolved through three distinct phases, each expanding the scope of what the orchestration layer is responsible for. In the first phase, the harness serves primarily as an interface between the model and the environment. In the second phase, it becomes the execution layer for developer assistants operating within persistent technical workspaces. In the third phase, it functions as the backbone of personal assistants that maintain continuity with individual users over time and across communication channels.

2.2.1 Action Interface: Connecting Model to Environment

The foundational layer of the harness establishes the connection between a model and its external environment. At this layer, the harness links the model with external tools such as file systems [78], browsers [79], and executable runtimes [80], translating outputs into executable actions and routing environmental results back for the next reasoning step. This closed-loop structure elevates a foundation model from a passive text generator to an agent capable of acting upon its environment [81].

The core mechanism is a repeating cycle of reasoning, action, and observation: the model generates an action, observes the outcome, and adjusts its next step based on the updated environmental state [82, 80]. A representative example is ReAct [82], which makes this loop explicit by directly interleaving reasoning traces with environment interactions. Subsequent work extends the same principle by incorporating program execution [80], API calls [25], and diverse external tools [83]. More recently, agent systems have organized these interactions within unified runtimes, enabling actions and observations to be handled consistently across different tools and environments [84, 78].

2.2.2 Workflow Infrastructure: Orchestrating Persistent Workspaces

Building upon the action interface, the second layer introduces persistent workspace management and multi-step workflow orchestration. The harness at this layer no longer merely bridges the model and individual tools; instead, it manages workspace state, coordinates sequences of development actions, and carries execution context across steps, enabling the model to inspect code [7], apply edits [85], run commands [86], and use results to guide subsequent actions [8]. This closed-loop development cycle elevates the model from a one-shot code generator to a developer assistant capable of iteratively refining its output until the task is complete [87].

The core mechanism at this layer is a repeating cycle of generation, execution, and revision: the model produces code, the harness executes it within the workspace, observes outcomes such as test results or error messages, and feeds these back for the next iteration [74, 88]. A representative example is SWE-agent [7], which demonstrates that strong task performance depends not only on code generation quality but critically on how the harness structures navigation, feedback, and iterative correction. Systems such as OpenHands [74] and Claude Code [88] further extend this principle by placing the model within realistic development environments with full access to shells, file systems, and version control, supporting sustained cycles of editing and validation rather than isolated generation. Beyond software engineering, LLM-in-Sandbox [26] shows that the same workspace infrastructure, virtualized as a minimal code sandbox, brings consistent gains across general non-coding tasks.

2.2.3 User-Centric Persistence: Continuity Across Sessions and Channels

The third layer extends the harness beyond single-session task execution by introducing continuity with individual users across sessions, tasks, and communication channels. Where the previous layer maintains workspace state within a task, this layer manages user history, preferences, routines, and ongoing tasks as persistent state, enabling personalized assistance that builds upon the prior interactions [89, 90]. This longitudinal interaction loop elevates the model from a stateless task executor to a personal assistant capable of sustained, context-aware engagement over time [91, 92].

The defining requirement at this layer is continuity with the same user over time, even when tasks span different sessions or communication channels [93, 90]. This enables longer-term, personalized assistance that accumulates knowledge of user needs and adapt accordingly. A representative example is OpenClaw [94], which treats the assistant as a persistent, user-oriented agent with support for memory, profile management, and delegated action across realistic tools and long-horizon interactions.

2.3 Connections with Related Concepts

Harness engineering is an emerging concept that intersects with several established notions in agent systems research. In this subsection, we clarify how harness engineering connects to and builds upon these related concepts.

Agent Framework. The concept most closely connected to harness engineering is the agent framework. Such frameworks provide reusable infrastructure for building, deploying, and managing agentic applications, typically covering tool interfaces, execution environments, interaction protocols, and support for multi-step task completion [76, 84]. Their core contribution is a concrete implementation foundation that turns agentic behavior into a working process. While the two concepts share common ground, they operate at different levels and address fundamentally different questions. Agent frameworks focus on the engineering question of *how to implement* an agent system — that is, realizing designs through concrete code and software architecture. Harness engineering, by contrast, addresses the methodological question of *how to systematically optimize* the overall performance of an agent system, encompassing not only improvements to the scaffold but also adaptations of the model itself. These optimization strategies are general across different frameworks and models and do not depend on the implementation details of any specific framework. An agent framework can therefore be seen as one concrete instantiation of a harness, while harness engineering serves as a meta-level discipline that guides the design of such instantiations. In this paper, we adopt this broader view and systematically examine design principles for building more effective agent systems through two complementary directions: scaffold optimization and model optimization.

Other Related Concepts. Beyond agent frameworks, three other families of concepts are closely related to harness engineering, each contributing an essential ingredient that harness engineering integrates into a unified system-level methodology.

• *Prompt engineering* and *context engineering* focus on crafting or structuring inputs to elicit better responses from LLMs [95, 96, 97]. Harness engineering subsumes these techniques as part of its model-side optimization, while further organizing multi-step execution, state management, and control flow across the entire agent workflow.

• *Tool use* examines whether and how LLMs can invoke external functions or APIs [98, 99]. Harness engineering incorporates tool use as a component within its scaffold design, additionally addressing when to invoke a tool, how to handle failures and integrate observations into subsequent reasoning.

• *Developer assistants* and *personal assistants* categorize agentic systems by their application roles (*e.g.,* coding support or everyday task assistance [74, 89]). Harness engineering provides the underlying orchestration structures, including context management, progress tracking, and robust execution, that enable such assistant roles to function effectively.

3 The Design of the Harness

Having established the conceptual framework of harness engineering in Section 2, we now turn to the scaffold side: the architectural components that wrap around the foundation model and shape its agentic behavior. Rather than operating as isolated modules, these components form an integrated pipeline. The agent workflow orchestrates perception, planning, and action in a closed loop; memory systems extend the agent's effective horizon beyond a single context window; skill libraries accumulate reusable behavioral units; and multi-agent orchestration scales the system to tasks that exceed the capacity of any single model instance. We examine each component in turn, focusing on design principles that have proven effective in deployed systems.

3.1 Agent Workflow

The agent workflow is the operational core of the agentic harness, transforming a stateless language model into a goal-directed system [100, 101]. Because a language model performs a single forward pass and terminates, what appears as persistent behavior is entirely constructed by the harness, which maintains message history and replays it on every invocation [102]. The harness compensates for this statelessness by wrapping the model in a loop: call the model, parse the response, execute the indicated tool, and feed the result back for the next iteration. This loop gives rise to three tightly

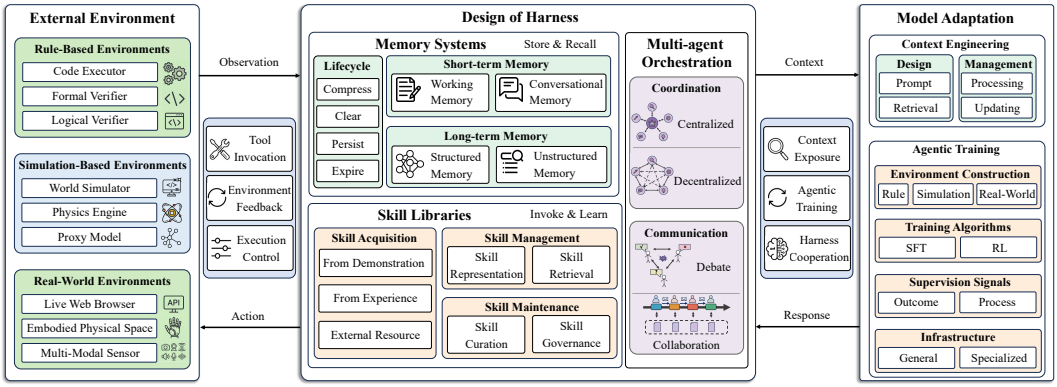

Figure 4: Overview of the harness design space and its interaction with external environments and model adaptation. **Left:** External environments provide grounding signals for agent execution and learning. **Center:** The four principal scaffold-side components that collectively form the harness infrastructure. The agent workflow governs the perceive-plan-act execution cycle; memory systems maintain persistent context across steps and sessions; skill libraries accumulate reusable capabilities to prevent redundant exploration; and multi-agent orchestration decomposes complex tasks across specialized agents. **Right:** Model-side adaptation through context engineering and agentic training, which complements the scaffold by shaping both the input information and the reasoning behavior of the underlying model. Together, scaffold-side construction and model-side adaptation form the two complementary pillars of harness engineering, and their synergistic optimization jointly shapes overall agent capability.

coupled phases. The harness first perceives the environment to gather task-relevant information (Section 3.1.1), then plans the next step based on accumulated state (Section 3.1.2), and finally executes the chosen action whose outcome becomes the input for the next iteration (Section 3.1.3).

3.1.1 Environment Perception

Environment perception is the entry point of each loop iteration. Before an agent can plan or act, the harness must construct a representation of the current environment that is both accurate and compact enough to fit within the context budget. This involves two complementary challenges: extracting task-relevant information from the raw environment, and encoding it into a form that downstream planning can reliably consume.

Observation Extraction. The central challenge is extracting spatial and logical relationships from complex interfaces, whether web pages [103, 104], desktop applications [105, 106], or mobile apps [69]. Early approaches extract structured features from UI hierarchies and convert them into textual representations [107, 108]. These representations are straightforward to parse but often large and redundant, motivating compression techniques that prune task-irrelevant content while preserving interactive elements [109]. Purely textual representations lack visual grounding and fail when layouts deviate from expected hierarchies [110]. Visual extraction based on screenshots addresses this limitation, with further gains from pretraining on large-scale GUI corpora [111] and reinforcement learning on interaction traces [112, 113]. However, purely visual methods still suffer from spatial ambiguity. The current state of practice therefore combines both channels through multimodal extraction, parsing structured metadata alongside screenshots to obtain element-level descriptions including widget type, text content, and spatial attributes [114]. This progression from textual to visual to multimodal extraction is not simply a march toward richer input; each step introduces its own cost. It reflects a fundamental trade-off among information completeness, token efficiency, and extraction reliability. Each additional perceptual channel provides a more complete picture of the environment but consumes more of the context budget, while every conversion step introduces opportunities for noise. Richer perception does not always yield better downstream decisions; the engineering challenge lies in maximizing useful signal per token for a given task.

State Representation. Once information has been extracted, the harness must encode it into a representation the model can use for planning. Common approaches employ structured markup

such as JSON [115], DOM-style trees [7, 108], or code formats [116] to produce compact textual descriptions. When these descriptions grow long, they exceed the context window or degrade planning quality through information overload [117]. Simple heuristics such as masking irrelevant attributes, truncating beyond a token budget, or retaining only recent observations can manage this growth effectively [109, 118]. Beyond such surface-level pruning, some methods use a secondary model to summarize or filter environment state [119], though these approaches do not capture how elements relate to one another. To model such relations, recent work distills the full environment into compact, task-oriented world models that retain only relevant objects, properties, and transitions [120], or builds explicit structures such as scene graphs [121, 122] that compress the environment into a small set of entities and spatial configurations most useful for planning. Industrial practice takes this filtering idea further and treats it as a core design principle: the perceptual interface should be designed as an Agent-Computer Interface rather than a Human-Computer Interface [123]. The harness surfaces only the few signals that materially affect decisions in the context window, while writing verbose details to files that the agent can query on demand. This ensures the context budget is spent on tokens with the highest decision-making value.

3.1.2 Task Planning

Task planning determines how the harness decomposes goals and maintains the plan state across loop iterations. Complex tasks cannot be solved in a single inference step because of two coupled constraints: autoregressive error accumulation causes success rates to drop sharply once reasoning depth exceeds what the model can sustain in one pass, and the finite context window cannot simultaneously hold the goal, full environment state, and all relevant constraints. Task decomposition addresses both constraints by segmenting a long reasoning chain into shorter chains separated by calibration points, where each new chain begins from a state confirmed through tool execution rather than from an unverified intermediate token. Finer decomposition inserts more calibration points and raises end-to-end reliability, at the cost of additional loop iterations and greater total token consumption. Existing work addresses this from two perspectives: how a complex goal is split into intermediate subtasks, and how those subtasks are converted into executable action sequences.

Task Decomposition. Early work organized intermediate reasoning into increasingly flexible structures, progressing from linear chains [95, 124] to branching trees [125] and graphs that allow path merging and reuse [126]. These approaches rely on predefined reasoning structures not adapted to specific tasks. More recent methods [127, 128] mitigate this mismatch by allowing agents to analyze the task and select suitable reasoning strategies before execution begins. Two emerging directions move beyond static structures altogether. The first is cross-session incremental decomposition [129]. When a task spans multiple context windows, single-pass planning breaks down because no single session holds the full problem state. A practical solution has the initial agent produce a complete functional requirements document that subsequent agents use to recover planning state, together with intermediate artifacts and version control history. Each new context window introduces a fresh agent that rebuilds task understanding from this externalized state before continuing where the previous session left off. The second is adaptive decomposition driven by external verifiers [123]. For tasks that resist natural parallelization, a known-good oracle can serve as a reference point. In one deployed example, multiple agents collaborate to build a C compiler; the harness assigns source file subsets to the agent-built compiler, compiles the rest with GCC, and cross-compares outputs to detect discrepancies. This turns an inherently sequential task into parallelizable subtasks whose boundaries are determined dynamically by runtime feedback rather than fixed in advance.

Plan Generation. Beyond decomposition, plan quality improves when each step incorporates explicit tool-use specifications [130] and when linear sequences are replaced with dependency-aware structures that enable independent subtasks to run in parallel [23]. The representation format of a plan also affects loop efficiency. A plan described in natural language may consume hundreds of tokens, whereas the same plan expressed as structured tool names and arguments requires far fewer. However, compressing too aggressively strips away the intent and assumptions behind each step, noticeably degrading the quality of retry attempts when replanning is needed after a failure [129]. This tension suggests that plan generation is best treated not as a one-time output but as a living artifact that the harness loop continuously verifies and adjusts.

3.1.3 Action Execution

Once a plan has been generated, the agent must translate it into concrete operations that affect the external environment. We examine action execution from two perspectives: how agents select and call discrete external tools, and how they operate within dynamic, stateful environments.

Tool Invocation. Tool invocation concerns how the harness enables an agent to select and call the appropriate tool for a given task. Early work [25] embedded tool tokens directly into the model's vocabulary, supporting only sequential single-tool calls. Subsequent methods [131] parse agent outputs into executable function calls, enabling branching and multi-step composition. To ground these capabilities in deployed settings, practical systems [7, 132] expose components such as file systems and shell commands as structured tool interfaces with explicit schemas. A converging line of work demonstrates that reliable tool invocation must be learned through executable environment feedback rather than specified in advance. Methods that train agents via reinforcement learning using runtime execution outcomes as direct signals show consistent gains in both single-tool reliability and multi-step composition [133, 134, 135]. Beyond the invocation mechanism, the granularity of the tool interface is a key harness design choice. A general-purpose interface such as bash offers maximum flexibility but leaves the harness with little ability to enforce whitelists, validate arguments, or isolate permissions. Specialized interfaces with explicit schemas, such as those defined in the Model Context Protocol, trade flexibility for interface-level safety checks [136]. Similarly, domain-specific systems such as SquRL [137] provide specialized interfaces by abstracting task-oriented tools [138] from the target domain. The right balance depends on the security requirements of the deployment domain. A related scaling challenge arises when the number of available tools is in the hundreds. Under the conventional pattern, every tool result returns to the context window, and long tool chains accumulate substantial token overhead. Code Execution with MCP [139] addresses this by having the model write a single code block that invokes multiple tools in sequence within a sandbox; only the final result returns to the context, collapsing a multi-round tool chain into a single model call.

Environment Interaction. Environment interaction concerns how agents execute actions in stateful environments whose consequences persist across steps. Early work [82] placed agents in raw environments with unbounded action spaces, making it difficult to reliably translate intentions into executable actions. Subsequent work [7] introduced structured agent-environment interfaces that expose available actions, expected formats, and resulting state changes. Building on this, filtering methods [140] combine the current plan with environment-based reward signals to discard infeasible or irrelevant actions before execution. Even with well-structured and filtered actions, reliable interaction requires that execution be isolated from the host system. Sandbox mechanisms [79, 78] confine agent behavior to controlled containers that restrict file system access, limit network exposure, and enable clean state resets between attempts. However, containment alone does not guarantee safety; recent analysis [141] shows that sandboxed agents still exhibit systematic vulnerabilities at planning, memory, and execution boundaries that require active trace-level auditing to detect.

Discussion

- The finite context window is a shared budget that perception, planning, and execution compete for: enriching one phase necessarily starves the others.
- The core value of agentic loops is not stepping through subtasks, but creating calibration points where external feedback corrects accumulated model drift.
- Loop reliability depends more on harness design; without proper support, agents compress all work into a single context window and prematurely terminate near the limit.

3.2 Memory Systems

Beyond the immediate execution loop, a capable harness must maintain agent coherence across reasoning steps that exceed the capacity of any single context window. This requires explicit memory systems that govern what to keep, what to compress, and what to offload. The guiding principle is not to remember as much as possible, but to persist only information that has reuse value for future decisions [142, 129]. We examine memory systems along these two dimensions: *short-term memory* (Section 3.2.1) manages information within a single session, while *long-term memory* (Section 3.2.2) accumulates knowledge that persists beyond the immediate context.

3.2.1 Short-term Memory

Short-term memory governs how an agent tracks and utilizes information within a single session. The harness must actively decide what intermediate state to preserve, compress, or discard within the context window. The key design insight is to distinguish between reconstructible information, such as tool outputs that can be re-fetched and are thus safe to discard, and irreplaceable information, such as reasoning conclusions and user decisions that must be retained or compressed [142]. We discuss two complementary aspects: *working memory*, which manages intermediate states during active reasoning, and *conversational memory*, which maintains dialogue history across turns.

Working Memory. Working memory provides a dynamic store for intermediate states generated during task execution. Early designs relied on linear traces that interleave reasoning steps, actions, and observations [82, 143], but linear formats struggle with non-linear reasoning involving branching or backtracking, often leading to duplicated information [144] or loss of critical context [27]. Subsequent work introduced structured alternatives: graph-based methods [145] organize intermediate thoughts as connected nodes supporting branching and recombination, while workflow-based methods [146] further organize execution states through reusable structures. Beyond format, industrial practice reveals that not all intermediate states share the same lifecycle. Tool call results can often be re-fetched and should be cleared first; task progress checkpoints must persist to support cross-window resumption [142, 129]. This lifecycle-aware decomposition moves working memory management from uniform compression toward selective retention based on reconstructibility.

Conversational Memory. In long dialogues, important information from earlier turns, such as user preferences or prior decisions, can be lost as history grows beyond the context window. Early methods stored conversation history externally and used hierarchical buffers [147] or paging mechanisms [148] to retrieve relevant content on demand, but these operate at coarse granularity and can split related context apart [149]. Later work achieved finer-grained segmentation by partitioning history into functional units such as task state [150] and user profile [151], though still treating each segment in isolation. Recent studies [152, 153] recover cross-segment dependencies through explicit links between memory elements, while the most recent efforts [154, 155] move toward adaptive management that learns what to retain, update, or discard based on the evolving dialogue state. This adaptive approach has already been deployed at scale. Claude Code, for instance, uses compaction during long sessions to produce structured summaries that preserve key decisions while reclaiming context budget [142], demonstrating that the distinction between reconstructible and irreplaceable information can be operationalized in production systems.

3.2.2 Long-term Memory

While short-term memory manages the active context window, long-term memory enables agents to accumulate knowledge that persists across sessions. Since the model maintains no persistent state between invocations, all mechanisms for storing, indexing, and retrieving long-term knowledge must be implemented by the harness. From a functional perspective, long-term memory divides into three categories with distinct lifecycles [156, 157]: semantic memory (facts, user preferences, project knowledge), episodic memory (past interactions, failure and success trajectories), and procedural memory (task rules and workflows). Each category requires different writing strategies and retrieval mechanisms. Moreover, not all long-term memory should be written in real time; preferences that must take effect immediately belong to hot-path memory, while experience extraction and conflict resolution are better handled asynchronously after the task completes [157]. We examine two organizational forms: *structured memory*, which maintains knowledge in explicit relational formats, and *unstructured memory*, which stores information as flat text.

Structured Memory. Structured memory organizes accumulated knowledge in relational forms such as knowledge graphs [158] or explicitly linked entry sequences [31]. Early graph-based designs [159, 160] store knowledge as nodes and relationships as edges, enabling efficient traversal but typically accommodating only one information type. Later approaches [161, 162] unified multiple types within a single structure using node tags, yet still require a fixed schema specified in advance, limiting generalization to unseen tasks. More recent work [31, 29] removes this rigidity by allowing each entry to carry its own contextual description and link to others based on semantic similarity and usage patterns rather than predefined types.

Unstructured Memory. Real-world experience often exists as flat text that lacks explicit relational organization [163]. Early work [148, 30] stored free-form text and used similarity matching for retrieval, but lengthy entries consume substantial space and outdated information is difficult to update. A subsequent line of work [24, 164] makes memory explicit by preserving past interactions and reflections in natural language, making it easier to inspect, edit, and distill into reusable guidelines. However, these systems rely on hand-designed rules for deciding what to store or discard. Recent work [154, 165, 166, 167, 168] treats memory operations as learnable decisions rather than fixed heuristics, with extensions to long-horizon personalization [169] and on-demand memory expansion [170].

Looking forward, long-term memory is evolving from an internal component of individual agents into a shared infrastructure layer. Systems such as Mem0 [171] and OpenMemory [172] abstract memory into a portable personal context layer, exposing standardized interfaces through protocols like MCP so that different agents can access the same memory on demand. At the technical level, memory becomes a reusable service rather than something each framework implements independently. At the ownership level, local-first designs store data on the user's machine, making memory persist across ephemeral agent sessions and enabling cross-application context continuity. This architectural shift also blurs the boundary between memory and skills: when an agent repeatedly distills reusable procedures from episodic memory and packages them as executable instructions, memory transitions from episodic to procedural form and eventually becomes a callable skill [173]. In this sense, skill libraries, discussed in the next section, can be viewed as an advanced form of memory systems.

Discussion

- The central challenge of memory is no longer capacity but governance: unbounded accumulation introduces stale conflicts, retrieval degradation, and persistent behavioral corruption.
- Context budget should be treated as a first-class runtime resource to be actively monitored and defended, not a passive container the model fills until it overflows.
- Agent state is not homogeneous—conversation history, tool results, task checkpoints, and long-term knowledge differ in persistence and update frequency, and must be managed separately rather than mixed into one undifferentiated context.
- The harness should enforce a disciplined lifecycle: compaction and clearing handle intra-session bloat, while only cross-session knowledge that remains valuable enters the persistent layer; reusable procedures further graduate from episodic memory into callable skills.
- Memory is evolving from an internal module into a shared, portable infrastructure layer: agents are ephemeral, but memory is long-lived, enabling cross-application continuity with user-owned context [171, 173].

3.3 Skill Libraries

While agent workflows enable action and memory systems preserve state, neither prevents the agent from rediscovering solutions to previously solved problems. Skill libraries close this gap by persisting reusable behavioral units that the agent can invoke when similar situations recur [35, 174, 175]. Conceptually, skill libraries occupy a continuum with the memory systems discussed in Section 3.2. When an agent repeatedly extracts actionable procedures from episodic memory and encodes them as executable instructions, memory gradually transitions from episodic to procedural form [176]. In production, this transition is increasingly explicit: skills are authored, versioned, and distributed as self-contained packages of instructions, scripts, and resources [177, 178]. The scope of skill library design therefore extends beyond learning alone to encompass construction, runtime routing, and lifecycle governance [179, 180]. This section examines the three core problems that arise when building and operating such libraries: how skills are acquired (Section 3.3.1), how they are organized and retrieved at runtime (Section 3.3.2), and how the library is maintained as it grows (Section 3.3.3).

3.3.1 Skill Acquisition

Skill acquisition converts successful behavior into reusable capabilities. The harness intercepts successful trajectories, triggers extraction routines, and persists the results for later retrieval. The central difficulty is generalization: a behavior observed in one trajectory must be abstracted sufficiently to transfer beyond the original context. Existing approaches address this from three directions.

Learning from Demonstration. The most direct route is to derive skills from prior successful behavior [35, 174]. Early methods imitate tool-call logs [25, 181, 182, 183] or replay expert trajectories [184, 185], yielding skills that reproduce observed actions but generalize poorly [186]. Injecting full trajectories into context also introduces redundancy [187]. Subsequent work therefore decomposes trajectories into atomic units that can be recombined into higher-level capabilities [34, 188], or identifies reusable structures within prior records as seeds for broader skill creation [189]. More recent methods further integrate textual knowledge as structured guidance during composition [190, 191].

Learning from Experience. Rather than relying on external demonstrations, agents can acquire skills through their own environment interactions. Early reinforcement learning formulations [192, 193] produce task-specific policies that transfer poorly [194, 33]. Later work reframes the problem as discovering reusable behavioral structures from interaction [195, 196, 197, 198, 199, 200]: successful action sequences are packaged into skills that persist beyond the originating episode. Recent approaches further improve coverage by guiding exploration toward regions of the environment likely to yield novel, generalizable behaviors [201, 36, 202]. In an extreme case, Tool-R0 [203] removes the need for any predefined task specification, training tool-calling skills entirely through self-play with executable environment feedback as the sole learning signal.

Learning from External Resources. Skills need not originate from agent trajectories at all. Industrial practice increasingly constructs them from documentation, code repositories, and API definitions [204, 177]. Anthropic's Agent Skills framework [177, 205] defines skills as directory-based packages of instructions, scripts, and resources that agents discover and load at runtime. OpenAI's Skills API [178] extends this model with versioned bundles that can be uploaded, managed, and mounted in hosted environments. In these systems, skills function as versioned software artifacts with lifecycle management analogous to traditional packages.

3.3.2 Skill Management

Once a library contains a large number of skills, two organizational problems arise: how to represent skills for efficient access, and how to select the right skill at runtime without overwhelming the model's context [206].

Skill Representation. Early work [184, 35] encodes skills as continuous embeddings, enabling similarity-based indexing but sacrificing internal structure. Later approaches adopt programmatic formats such as executable code fragments [190], structured reasoning procedures [34, 195, 207], or unified tool-use schemas [208, 37], making composition and transfer more explicit [194]. Production systems take this further by representing skills as complete packages containing instructions, scripts, metadata, and tests [177, 178]. This package-level design enables progressive disclosure: the agent initially loads only metadata or a directory-level summary and expands into full instructions on demand [177, 209]. Because only a fraction of any skill occupies the context at a given time, the library can scale well beyond what full-injection approaches support [210, 179].

Skill Retrieval. Embedding-based similarity [195, 184] provides a natural method for retrieval but degrades when tasks require composing multiple skills, as in multi-agent coordination [211, 212] or embodied manipulation [213]. Hierarchical retrieval mitigates this by first resolving coarse intent and then searching within a narrower subset [208, 210]. Context-sensitive retrieval conditions selection on environmental states, prior trajectories, or intermediate failures [214, 215], turning retrieval into an iterative decision process [216]. At the production scale, where libraries contain thousands of overlapping entries, retrieval becomes a learned routing problem. SkillRouter [179, 206] demonstrates this with a retrieve-and-rerank pipeline that operates over full skill text, showing that exposing only names and descriptions degrades routing accuracy. This finding indicates that skill content itself is a critical routing signal and that dedicated learned routers outperform metadata-only matching.

3.3.3 Skill Maintenance

As libraries grow, the challenge shifts from acquisition and retrieval to long-term health. A static library inevitably accumulates redundant, outdated, or conflicting entries that degrade retrieval precision and execution reliability.

Library Curation. Skills decay in real environments as tool APIs change and external dependencies evolve [217, 218]. The harness must detect execution failures, trace them to specific skills, and trigger targeted updates. At the library level, growth introduces overlap and staleness that impair the retriever's ability to discriminate among candidates [179, 210]. Active curation, including merging redundant entries, pruning ineffective ones, and hierarchically organizing surviving skills, is therefore essential to sustained performance.

Skill Governance. When skills are treated as versioned executable packages [178, 219], they inherit the governance requirements of traditional software. New versions require correctness testing, and existing skills require regression checks as environments change [219]. Distribution across platforms further introduces supply-chain risks: incorrect or malicious skills can persistently corrupt agent behavior [220, 177]. Reliable operation at scale therefore demands version control, compatibility checking, conflict detection, and principled deprecation.

Discussion

- A fundamental tension persists between specificity and generality: trajectory-extracted skills tend to be narrow and brittle, while highly abstract skills lack the operational precision needed for reliable execution.
- Once skills become versioned artifacts, they inherit software lifecycle burdens—API changes, environment migration, and silent error propagation—that demand active maintenance, not just extraction.
- The boundary between memory and skills is blurring: repeatedly distilled episodic memory becomes indistinguishable from skills, suggesting future harnesses should unify experience accumulation, procedural extraction, and skill maintenance into a single governed pipeline.

3.4 Multi-agent Orchestration

Single-agent harnesses, however well designed, face inherent limits in parallelism, specialization, and fault tolerance. Multi-agent orchestration addresses these limits by coordinating multiple agents, where each runs its own workflow and memory, toward a shared task objective. Product practice identifies three axes along which multi-agent systems deliver value [221]: context protection, where each agent operates in an isolated context window; parallelization, which trades token consumption for shorter wall-clock time; and specialization, where splitting agents by domain avoids the performance degradation that occurs when a single agent is loaded with too many tools. This principle shapes the two central design questions explored below: *coordination architectures*, which define how agents are structurally organized, and *communication mechanisms*, which define how agents exchange information within these structures.

3.4.1 Coordination Architectures

Coordination architectures define how control authority, task assignment, and result aggregation are distributed across agents. Existing approaches divide by whether the system relies on a core agent: *centralized architectures* employ a controller to manage the entire system, while *decentralized architectures* achieve coordination through interaction among equal agents.

Centralized Architectures. Centralized architectures use one central agent to decompose tasks, assign subtasks to specialized subagents, and aggregate their results. Early systems such as MetaGPT [39] and ChatDev [40] organize role-specialized agents through manually designed workflows, but these patterns are costly to build and difficult to transfer across tasks. Subsequent work shifts toward autonomous coordination: AutoGen [38] treats scheduling as a configurable conversation and routing layer, while MegaAgent [222] enables agents to negotiate responsibilities at runtime, allowing organization to emerge without predefined orchestration [223, 22, 224, 225]. A particularly successful centralized pattern in production is the verification subagent [221], where a dedicated subagent tests or verifies the primary agent's work. This pattern succeeds because verification requires minimal context transfer: the verifier can black-box test the system without knowing its full construction history, avoiding the context degradation that occurs when reasoning passes between agents. However, scaling centralized systems is not as simple as adding more subagents. Recent findings [226] show that smaller teams with well-designed memory can outperform larger ones,

suggesting that the quality of each agent's coordination infrastructure matters more than team size alone.

Decentralized Architectures. Decentralized architectures [227] explore how collaboration can emerge from interactions among equal agents without a central coordinator. AgentVerse [41] demonstrated that agents can spontaneously form roles and divide labor through interaction alone, and DyLAN [228] extended this with a dynamic collaboration graph that evolves as the task progresses [42, 229, 230]. A known risk of this design is premature consensus: if agents agree too quickly, minority but correct viewpoints can be overlooked. CONSENSAGENT [231] addresses this by assigning asymmetric roles so that minority positions receive structured consideration before a final decision. Decentralized coordination does not always require a purpose-built communication protocol. In a large-scale multi-agent coding project [232], 16 agents coordinated without direct communication: each claimed a task by writing a file to a shared directory, and conflicts were surfaced through git merge conflicts. This demonstrates that existing software infrastructure, such as file systems and version control, can serve as the coordination substrate. The shared-state pattern [233] generalizes this: agents collaborate through a shared state space, building on each other's findings without a central router, which eliminates single-point failure but risks duplicate work and conflicting decisions.

3.4.2 Communication Mechanisms

Communication mechanisms address how agents exchange information within a multi-agent system. Without well-designed communication, even a well-structured coordination architecture degrades into disconnected model calls rather than a coherent collaborative system. The central challenge is deciding what information to share and at what granularity. Sharing only final results is insufficient, because actions carry implicit decisions, agents that lack the reasoning behind a result are likely to make conflicting choices [234]. Sharing full agent traces preserves maximal information but creates prohibitive communication overhead. A practical compromise is to compress action histories into summaries of key decisions and events, preserving information density while controlling cost [234]. Existing work addresses this challenge from two directions: *debate-based methods*, which improve output quality through critique and disagreement, and *collaboration-based methods*, which support coordinated execution across agents.

Debate-based Methods. Debate-based methods address a fundamental limitation of single agents: following one line of thought makes it difficult to uncover intermediate errors. A natural solution is to have multiple agents execute the same task from different roles [39, 235, 236] and stances [237, 238], so that agents challenge and refine each other's outputs before forming a final answer [239]. However, multi-agent discussion does not automatically lead to better results [240]. Without a mechanism to integrate conflicting viewpoints, interaction can become repetitive or unstable. More recent work therefore organizes debate around explicit reflection loops [241] or consensus-oriented decision procedures [231] that summarize conflicting viewpoints and assemble them into a coherent result.

Collaboration-based Methods Collaboration-based methods coordinate agents into a coherent execution structure where different roles contribute complementary capabilities [242, 243, 40]. Early work relies on explicit role specialization and predefined workflows [244, 39], making systems controllable but fragile under dynamic task changes [223]. Recent research shifts toward more adaptive approaches, including graph-based planning that restructures coordination as execution evolves [245, 246], reinforcement learning that jointly optimizes collaboration policies [247, 248, 249, 250], and environment-aware coordination that adjusts strategies based on real-time progress [251]. Another emerging direction is artifact-mediated collaboration, where agents coordinate through durable shared project state—such as permission-scoped workspaces containing analyses, plans, and code—rather than transient conversational handoffs [252]. A practical challenge for collaboration is the read-write asymmetry identified in production systems [253]: multi-agent systems that primarily read are far easier to build than those that primarily write. When multiple agents write simultaneously, their implicit decisions can conflict, producing incompatible outputs that are difficult to merge [234]. This asymmetry suggests that collaborative writing tasks require stronger coordination protocols than parallel reading tasks.

Table 1: Design principles of representative agent systems across the four harness components.

Components	Claude Code	OpenClaw	Hermes Agent
Agent Workflow	• Plan-then-execute separation • Layered permission control • Interactive user confirmation	• Declarative context assembly • Context-driven tool selection • Unified multi-channel routing	• Budget-capped execution loop • Security-aware context filtering • Platform-adaptive prompt design
Memory Systems	• Automatic context compaction • Hierarchical file persistence • Typed cross-session memory	• Vector-based semantic retrieval • Wiki-based linked knowledge • Automatic memory capture and recall	• Threshold-triggered compression • Dialogue-driven user profiling • Snapshot-based memory loading
Skill Libraries	• User-authored slash commands • Agent-assisted skill creation • Persistent scheduled tasks	• Centralized skill registry • Standardized skill interfaces • Role-based skill assignment	• Agent-managed lifecycle control • Automatic curation and retirement • Security-verified activation
Multi-Agent Orchestration	• Role-based specialist subagents • Repository-level workspace isolation • Foreground and background execution	• Depth-limited agent spawning • Identity-based agent configuration • Cross-framework communication protocol	• Explicit orchestrator role • Persistent cross-agent task board • Parallel multi-model execution

Discussion

• Each coordination pattern carries an inherent tension: centralized designs risk single-point bottlenecks, while decentralized designs increase the difficulty of debugging and behavior prediction [233].
• Decomposition should follow context boundaries, not problem types: splitting by role creates persistent coordination overhead, whereas the agent holding the context should own all related subtasks [221].
• A single agent failure can redirect the entire exploration trajectory, creating cascading errors.
• Getting a multi-agent system to basically work takes $\sim$20% of the effort, while making it reliable takes the remaining $\sim$80% [233].
• Multi-agent orchestration may be retracing the path of single-agent systems: manually designed coordination patterns may eventually be internalized as native model capabilities.

3.5 A Comparative View of Representative Agent Systems

To make the preceding design principles concrete, we now examine how three representative agent systems, Claude Code [88], OpenClaw [11], and Hermes Agent [12], instantiate the four harness components in practice. Each agent system embodies a distinct design philosophy, and Table 1 summarizes the key differences.

Agent Workflow. The three systems govern the perceive, plan, and act loop in fundamentally different ways. Claude Code enforces a strict plan-then-execute separation: the agent first enters a read-only phase to explore the codebase and draft a structured plan, then requests interactive user confirmation before any modification is carried out, and every subsequent tool call passes through a layered permission control mechanism so that no consequential action proceeds without explicit consent. OpenClaw treats the loop as composable infrastructure, where declarative context assembly specifies how information enters, persists in, and leaves the context window, context-driven tool selection prunes the available tool set before each model call based on runtime signals such as authentication state or channel type, and unified multi-channel routing allows a single agent to serve dozens of messaging platforms through one dispatch interface. Hermes Agent emphasizes autonomous self-governance, implementing a budget-capped execution loop that bounds each task with an explicit step limit, security-aware context filtering that scans every loaded file for prompt-injection threats before it reaches the model, and platform-adaptive prompt design that tailors system instructions to the runtime environment, so that the agent can operate reliably without human oversight.

Memory Systems. The three systems differ sharply in how they manage working memory and long-term knowledge. Claude Code is context-window-centric: the conversation itself serves as working memory, and once it grows too long the system performs automatic context compaction by compressing it into a structured summary; long-term knowledge resides in hierarchical file persistence at the user, project, and directory level that is loaded automatically into every session, supplemented by a typed cross-session memory store that organizes persistent facts by category such as user preferences, feedback, and project state. OpenClaw adopts a dual-track design that pairs vector-based semantic retrieval, which surfaces relevant facts before each model call, with wiki-based linked knowledge that maintains human-readable, interlinked persistent articles, both underpinned by automatic memory capture and recall that records notable facts and retrieves them without explicit user commands. Hermes Agent focuses on streaming context management: threshold-triggered compression monitors token usage in real time and compacts the context once a configurable limit is crossed, while protecting the earliest and most recent messages so that task framing and recent state survive; for long-term user modeling it employs dialogue-driven user profiling through an external service that maintains an evolving user profile by reasoning over past conversations, and it adopts snapshot-based memory loading that ingests memory files as frozen snapshots at session start so that mid-session updates do not invalidate the prompt cache.

Skill Libraries. The three systems organize reusable capabilities at increasing levels of autonomy. Claude Code treats skills as user-authored slash commands, where each skill is a markdown file with structured metadata; a built-in helper enables agent-assisted skill creation so that the system can bootstrap new skills on its own, and persistent scheduled tasks let recurring jobs behave like durable skills, all governed by plugin blocklists and fine-grained permission allowlists. OpenClaw moves to a centralized skill registry that packages each skill with standardized skill interfaces and publishes them through a hub for discovery and installation, with role-based skill assignment ensuring that each agent only receives skills relevant to its designated role, cleanly separating skill authoring from consumption. Hermes Agent goes furthest by implementing agent-managed lifecycle control: skills are exposed through progressive disclosure in three tiers of increasing detail so that the agent loads only what it needs, automatic curation and retirement removes skills that have not been invoked within a configurable period, and security-verified activation validates every skill before it is enabled.

Multi-Agent Orchestration. All three systems support subagent delegation but differ in coordination philosophy. Claude Code follows a pattern of role-based specialist subagents, spawning agents with explicit role annotations such as explorer, planner, or general-purpose, each receiving a constrained tool set; repository-level workspace isolation gives each subagent its own working copy, and the option of foreground and background execution keeps each subagent in a clean context window of its own. OpenClaw takes a protocol-first stance, with depth-limited agent spawning that bounds recursion to prevent runaway chains, identity-based agent configuration that defines each agent's role, personality, and behavioral constraints through scoped configuration files, and a cross-framework communication protocol that enables agents built on different stacks to interoperate as a first-class concern. Hermes Agent offers the richest coordination toolkit by defining an explicit orchestrator role that separates coordinating agents from leaf agents that execute tasks, maintaining a persistent cross-agent task board for shared work tracking, and most distinctively supporting parallel multi-model execution that queries multiple language models simultaneously and synthesizes their outputs.

Discussion

- The same four harness components can be instantiated under very different design philosophies: Claude Code optimizes for human-in-the-loop safety, OpenClaw for composable extensibility, and Hermes Agent for autonomous self-governance.
- These trade-offs directly reflect deployment context: interactive developer assistance, multi-channel platform orchestration, and autonomous terminal execution demand fundamentally different balances of control, flexibility, and autonomy.

4 Model Adaptation for Harness

Within the synergistic framework of harness engineering, designing the surrounding harness must be complemented by model-side adaptation. Standard LLMs are trained for general text generation

and lack the built-in capacity to invoke tools, follow multi-step workflows, or process environmental feedback. We organize model adaptation into two categories: *Context Engineering* (Section 4.1), which shapes model behavior at inference time through the information exposed by the harness, and *Agentic Training* (Section 4.2), which internalizes agent-specific behaviors into model parameters through imitation and reinforcement learning.

4.1 Context Engineering

The contextual information provided during inference determines both the behavior and the capability ceiling of an LLM agent. As discussed in Section 3, the harness governs what the model observes at each step, including task instructions, tool descriptions, retrieved knowledge, and interaction history. Context engineering adapts model behavior by deciding what information to expose, how to structure it, and how to update it throughout multi-turn interactions. Unlike general prompting, context engineering for agentic harnesses specifically targets tool use, workflow execution, and long-horizon decision making under limited context budgets. We cover two complementary stages: *Context Design*, which constructs high-quality inputs before the model acts, and *Context Management*, which compresses and maintains contextual state over extended interactions.

4.1.1 Context Design

Context design focuses on constructing optimal input before an agent reasons or acts. This relies on two core strategies: *Prompt Engineering*, which structures instructions provided to the model, and *Context Retrieval*, which expands the model's knowledge by incorporating external information.

Prompt Engineering. Prompt engineering steers model behavior through carefully crafted inputs without parameter updates. Early work focuses on structuring task instructions and demonstrations [254, 255, 125, 126], while subsequent studies show that explicit role descriptions can effectively elicit targeted behaviors [256, 227]. This proves particularly useful in multi-agent systems where agents assume specialized roles and coordinate within shared workflows [159, 45]. A common limitation of manual prompt design is its heavy reliance on human effort. To address this, automated prompt optimization enables models to refine their own prompts. APE [16] uses the LLM itself to generate and select candidate instructions, while subsequent methods eliminate human intervention entirely through self-refining loops [257, 258, 259].

Context Retrieval. Context retrieval dynamically integrates external knowledge to overcome the limitations of parametric memory. Traditional retrieval-augmented generation (RAG) relies on a static retrieve-once paradigm with flat text chunks, which struggles with complex multi-hop queries [260]. Subsequent methods advance retrieval along two dimensions: *retrieval timing* and *knowledge structure*. For retrieval timing, active mechanisms allow models to interleave reasoning with information gathering, dynamically deciding when to retrieve and how to rewrite queries based on intermediate states [261, 262, 263, 264, 265, 266, 44]. For knowledge structure, graph- or table-based indexes capture explicit relations among retrieved items [267, 268, 269, 43]. Building on these advances, agentic retrieval integrates retrieval operations directly into the model's iterative execution loop. Pioneered by ReAct [82], this paradigm enables agents to alternate between reasoning and querying. Subsequent frameworks further abstract retrieval into automatically constructed modules [270] and hierarchical strategies [271], while multi-agent ecosystems distribute retrieval across specialized roles to handle complex tasks through collaborative reasoning [272].

4.1.2 Context Management

Raw context is often noisy and excessively long, risking both context overflow and reasoning degradation [273]. Context management involves two processes: *Context Processing*, which compresses information for immediate inference, and *Context Updating*, which maintains accumulated experience across interactions.

Context Processing. Context processing compresses the assembled context to maintain maximal information density within the model's limited window. Early methods shorten input text by pruning tokens based on importance [274, 275, 276]. While effective for single-turn queries, these methods overlook the sequential dynamics of autonomous agents. To prevent context overflow in long-

horizon tasks, recent frameworks design compression specifically for multi-turn agent trajectories. Acon [277] applies reinforcement learning over interaction histories to minimize context usage while maintaining long-term memory. AgentDiet [46] prunes redundant information to reduce inference costs, AgentFold [278] condenses sub-tasks through hierarchical consolidation, and SWE-Pruner [279] performs task-aware adaptive pruning for coding agents. Beyond pruning existing traces, IterResearch [280, 281] reframes context management as workspace reconstruction, replacing full-history replay with a compact state consisting of the original question, an evolving report, and the latest action-observation pair.

Context Updating. Context updating dynamically maintains and evolves an agent's contextual state across extended interactions, deciding what to remember, what to forget, and how to reorganize experience. At the foundational level, agents accumulate experience through reflection. Reflexion [24] pioneers this by having agents summarize failure causes for future attempts, while AWM [146] distills reusable workflow patterns from successful trajectories. As experience grows, flat storage becomes impractical, prompting hierarchical memory architectures [148] that partition memory into a fast context window and slow external storage with autonomous page-swapping. Subsequent frameworks enhance this with graph databases [29], streaming caches [282], and time-decay forgetting [30]. However, passive storage alone cannot support active memory lifecycle management. Recent studies empower agents to organize interconnected knowledge [32, 283, 284], highlight important memories [31], and delete outdated information [285, 286].

4.2 Agentic Training

While context engineering adapts model behavior at inference time, its effectiveness is bounded by the model's inherent capabilities since parameters remain fixed. Agentic training complements this by internalizing domain knowledge and agent-specific behaviors directly into model parameters through two broad paradigms: *supervised fine-tuning* (SFT), which learns from demonstrations or teacher-generated trajectories, and *reinforcement learning* (RL), which refines policies through environment interaction feedback. Applying these methods to agents introduces unique challenges: constructing interactive environments, assigning credit under sparse rewards, stabilizing policy optimization, and managing large-scale engineering overhead. We discuss four aspects: *Environment Construction* (Section 4.2.1) explores how to build interactive training environments; *Reward Design* (Section 4.2.3) examines how to acquire stable reward signals; *Training Optimization Algorithms* (Section 4.2.2) discusses both SFT-based and RL-based policy updates; and *Infrastructure* (Section 4.2.4) discusses systems for large-scale distributed training.

4.2.1 Environment Construction

Training environments form the foundation of agentic training. Since the harness mediates interaction between the model and the external world, environment fidelity determines whether learned policies transfer to real-world deployment. Existing environments fall into three types based on feedback mechanism: *rule-based environments* provide deterministic binary rewards from execution outcomes; *simulation-based environments* use proxy models to approximate real-world dynamics; and *real-world environments* place agents directly in live systems with authentic feedback.

Rule-Based Environments. Rule-based environments have evolved from predefined sandboxes to strictly verifiable metrics. Early studies [287] train agents in rule-based sandboxes with predefined state transitions, supporting household [288] or e-commerce [289] scenarios. However, predefined actions require manual design and struggle to generalize to complex reasoning with large action spaces. Recent work [290, 291, 292, 293] demonstrates that with reliable binary outcome feedback, agents can continuously self-improve without human annotations. This principle is rapidly adopted in mathematical [294, 295, 10] and coding [296, 132] domains where precise verifiers provide reliable feedback. More recent environments further extend this through procedurally generated reasoning tasks [297] and stateful enterprise sandboxes supporting long-horizon planning [298]. In software engineering, Scale-SWE [299] scales environment construction by turning raw GitHub pull requests into verified Dockerized SWE tasks and training trajectories via a sandboxed multi-agent workflow.

Simulation-Based Environments. Simulation-based environments use proxy models to provide low-cost feedback approximating real-world dynamics. A critical challenge is that LLM-as-simulator

approaches are susceptible to hallucinations in physical scenarios. To improve fidelity, some approaches [300] train world models on large-scale interaction data for cross-domain generalization, while neural simulators extend this to operating-system interfaces by modeling screen transitions [301]. Beyond fidelity, improving rollout efficiency is equally important. Hardware-accelerated physics engines support massively parallel multi-agent rollouts [302, 303], while RAP [304] shifts simulation inside the model itself, allowing it to compare possible outcomes before acting. To further improve accuracy, recent works [305] replace generative proxies with deterministic database-backed mechanisms that execute actions as SQL transactions.

Real-World Environments. Real-world environments place agents directly into live digital or physical systems with authentic, constantly changing feedback. Early research [108, 306, 307] used controlled static websites, but these lack realism. Later work introduced actual digital noise such as network delays and dynamic content loading [113, 308, 309, 310]. To further scale up, some studies [311, 312, 313] let agents freely browse the live web using real-time operational logs as feedback. The physical world is the most complex environment because robotic actions must follow real physics and obey strict safety rules. Embodied environments [314, 315, 316] combine live sensor data with physics-based safety checks, enabling agents to translate high-level reasoning into real physical movements.

4.2.2 Training Optimization Algorithms

After constructing interaction environments and reward signals, agentic training requires optimization algorithms to update model parameters. Existing methods follow two broad paradigms: *supervised fine-tuning*, which learns agentic behaviors by imitating demonstrations or self-generated trajectories, and *reinforcement learning*, which refines policies through trial-and-error interaction with environment feedback.

Supervised Fine-Tuning. Supervised fine-tuning transfers agentic capabilities into the target model through direct optimization on multi-turn trajectories containing intermediate plans, tool calls, and environment observations. Based on data origin, methods are divided into off-policy distillation from teacher models and on-policy generation from the student itself. Off-policy distillation trains on trajectories pre-generated by stronger models. FireAct [317] first applies this to agentic scenarios, showing that fine-tuned small models can outperform larger ones on tool-use tasks. Since agent-only data may degrade general capabilities, later studies improve data quality through trajectory-instruction mixing [318], format and reasoning signal disentanglement [319], and unified multi-turn formatting [320]. To improve scalability, recent work synthesizes training data through environment interaction and systematic data engineering [119, 321, 322, 323, 299]. Within this line of work, ClawGym [324] introduces a scalable synthesis-centered framework for Claw-style personal agents, generating workspace-grounded tasks and further linking them to agent training and diagnostic benchmark construction. When trajectories contain both planning and execution signals, multi-teacher supervision [325, 326] or segment-specific losses [327] can separate these components for better transfer. On-policy distillation addresses the train-test distribution mismatch by letting the student generate its own trajectories while the teacher provides dense supervision. Early methods optimize divergence objectives such as forward KL [328] and reverse KL [329]. To stabilize training under large capability gaps, subsequent methods adjust target logits [330] or clip reward signals [331]. Beyond imitation, ExOPD [332] combines distillation with KL-regularized RL to push the student beyond the teacher, while GAD [333] extends on-policy distillation to black-box teachers using discriminator-based feedback.

Reinforcement Learning Approaches. While supervised fine-tuning internalizes agentic behaviors, it cannot support exploration beyond the training distribution. Reinforcement learning addresses this by refining policies through trial-and-error with reward feedback. Based on whether a value network is used, methods are divided into critic-based and critic-free algorithms. Critic-based algorithms introduce a value network to estimate advantages and guide stable policy updates. PPO [334] is the representative method, with improvements including value pre-training [335] and reward normalization [336, 337]. However, standard PPO struggles with long-horizon agents because credit assignment across multi-step trajectories is inherently difficult. Recent methods address this by decomposing interactions into planning levels [338, 339] or reasoning trees [340], and by assigning separate critics to different agents in multi-agent settings [341]. Critic-free algorithms remove the

value network to reduce memory and computation costs. One line uses offline preference learning such as DPO [342], KTO [343], SimPO [344], and ORPO [345] to avoid expensive online exploration. Another preserves online exploration but estimates advantages directly from rewards: GRPO [346] and ReMax [347] use parameter-free estimators based on group-level or maximum-return signals. For agentic workloads, recent studies further introduce hierarchical sampling and tree-structured exploration [50, 348, 349], token-level loss weighting [48], dynamic sampling [350, 351], and entropy balancing [351, 352, 353] to stabilize long-horizon exploration.

4.2.3 Supervision Signals

After environments are constructed, reward signals evaluate agent behaviors and guide policy improvement. Based on supervision granularity, two complementary paradigms emerge: *outcome-level signals* provide feedback only upon task completion, while *process-level signals* offer dense supervision after each step.

Outcome-Level Signals. Outcome-level signals evaluate the final state of a task, working well in domains that are easy to verify but hard to solve. In mathematical [354, 355], code generation [356, 357], software engineering [358], and deep research [49, 312, 359] tasks, objective execution feedback serves as a natural outcome signal. A common approach [358, 360] uses binary signals to let models learn from both correct and incorrect trajectories, driving large-scale self-play and iterative repair. To further reduce reliance on human labels, TTRL [361] estimates rewards through majority voting over sampled solutions, enabling RL without gold annotations. In formal theorem proving where signals are extremely sparse, DeepSeek-Prover-v1.5 [362] combines proof assistant feedback with tree-based search and exploration rewards.

Process-Level Rewards. Outcome-level rewards suffer from signal sparsity in long-horizon tasks, as models cannot determine which specific steps lead to success or failure. Process Reward Models (PRMs) address this by providing dense supervision at each reasoning or action step. The evolution of PRMs involves two aspects: *supervision signal construction* and *architectural design*. For signal construction, early methods [363] relied on expensive human annotations. Subsequent work automated this through Monte Carlo sampling [364] and Monte Carlo Tree Search [365]. Recent extensions introduce continuous spatial rewards for GUI grounding [366] and fine-grained rewards for tool selection [51]. For architectural design, traditional PRMs are discriminative models that assign a single score per step [363], but they are difficult to interpret and prone to bias. Generative methods [367] address this by performing step verification through natural language reasoning, using chain-of-thought to explain correctness, yielding more transparent reward signals. Complementary to reward-based optimization, supervised fine-tuning on curated trajectories offers a more direct signal: high-quality trajectories are sourced from human specialists [368] or distilled from strong models via rejection sampling [258], with many systems combining both to bootstrap capability [4].

4.2.4 Infrastructure

Large-scale agentic training requires coordinated scheduling of model inference, environment interaction, gradient computation, and parameter synchronization. The efficiency of the system infrastructure directly limits the speed and scalability of optimization.

General-Purpose Frameworks. General-purpose agentic frameworks coordinate the core RL components (actor, critic, reward, and reference models) across distributed clusters while supporting multi-turn interactions and continuous environment feedback. For flexible scheduling, some systems [369, 370, 47, 371] dynamically assign models to specific GPU partitions, optimizing data transfer across training stages. To improve throughput, asynchronous architectures [372, 373] separate the slower generation stage from gradient updates, scaling efficiently to thousands of GPUs. For environment integration, standardized APIs [374, 375] wrap various simulators into unified interfaces for cross-environment training. However, the multi-turn nature of agents causes load imbalance among fast generation, slow environment interaction, and heavy gradient updates. Recent frameworks address this by either interleaving generation and environment execution within a single workflow [50, 348, 376, 377, 378, 379], or physically separating stages across machines [380, 381].

Specialized Frameworks. Specialized frameworks extend general-purpose infrastructure for advanced settings such as multi-agent coordination and omni-modal interaction. For multi-agent systems, multiple agents interact through long-horizon interdependent trajectories, making rollout scheduling far harder than in single-agent settings [382]. Dedicated frameworks [383, 382, 341, 384, 385] introduce centralized multi-agent interaction, distributed policy training, and asynchronous multi-turn rollouts. For omni-modal agentic training, heterogeneous observations and feedback across text, images, and videos pose unique challenges [386]. Recent studies introduce specialized rubric rewards for multimodal grounding [386, 387] and multimodal search environments [388], extending agentic training beyond standard textual single-agent settings.

5 Harness Capacity and Evaluation

In this part, we review representative benchmarks and evaluation methods for agentic harnesses. The capability of an agentic system depends not only on the underlying model but equally on the harness that organizes context, tools, execution, and feedback. We refer to this overall ability, which sustains effective behavior under tool use, partial observability, long execution trajectories, and changing environmental feedback, as *harness capacity*. Measuring harness capacity requires going beyond single-step correctness; evaluation must assess whether a system can maintain goal-directed progress across multi-step interaction with external environments. Recent benchmarks embody this perspective by situating agents in increasingly realistic environments, extending task horizons and statefulness, and incorporating richer evaluation signals including execution results, environment states, rubric-based judgments, and run-to-run reliability [108, 389, 105, 390, 132, 53, 62, 60, 73, 391, 392, 59, 393, 68, 394]. Benchmark design has thus become an integral part of harness engineering itself, as it determines which aspects of orchestration—context management, tool coordination, error recovery—are made visible and measurable.

5.1 Deep Research

Deep research benchmarks evaluate whether an agent can support open-ended information seeking beyond short-answer retrieval. The field has progressed from difficult browsing questions to broader search, citation-grounded reporting, and rubric-based evaluation of synthesis quality.

Broad and Complex Search. An early direction focuses on information-heavy search rather than short-answer lookup. BrowseComp and WideSearch evaluate whether an agent can sustain multi-step exploration, compare evidence across search branches, and preserve useful intermediate findings [52, 53, 54, 391]. LiveDRBench is particularly notable for characterizing deep research in terms of breadth, logical nesting, and search intensity rather than report length alone. Compared with classical retrieval settings, these benchmarks stress search planning, evidence accumulation, and the harness's ability to maintain coverage as the search space expands.

Research Reports. A second line of work shifts attention from information collection to citation-grounded long-form reporting. DeepResearch Bench, ReportBench, and LiveResearchBench evaluate whether an agent can organize retrieved evidence into coherent outputs whose claims, citations, and structure can be directly inspected [55, 395, 396, 392, 397, 398, 399]. This direction places greater weight on report organization, citation quality, and faithfulness between statements and sources, revealing how much performance depends on the harness's ability to manage notes, compress documents, and preserve grounding during generation.

Survey Generation and Evaluation. More recent benchmarks move toward survey-style synthesis with explicit quality evaluation. ResearchRubrics pairs deep-research prompts with expert-designed rubrics, making coverage, grounding, and analytical quality first-class evaluation targets [400, 392, 397]. This marks a shift from asking whether an agent can produce a plausible artifact to asking whether the synthesis is complete, well-supported, and genuinely useful under structured judgment. Even so, current deep-research benchmarks still under-evaluate source credibility, citation faithfulness under changing web content, cross-session persistence, and the cost-latency trade-offs that matter in practical research assistants.

Table 2: Representative benchmarks across major task domains in agentic harness engineering.

Benchmark	Date	Tasks	Evaluation Mechanism
		Deep Research	
BrowseComp	2025.04	Browsing-based information seeking	Rule-based
Deep Research Bench	2025.05	Web research	Rule-based
DeepResearch Bench	2025.06	Deep research report generation	Rule-based
ResearcherBench	2025.07	Frontier scientific inquiry	Rule-based / LLM Judge
ReportBench	2025.08	Academic survey / report generation	Rule-based
Characterizing Deep Research	2025.08	Deep research benchmarking	Rule-based
WideSearch	2025.08	Agentic broad information seeking	Rule-based
PDR-Bench	2025.09	Personalized deep research	LLM Judge
LiveResearchBench	2025.10	User-centric deep research in the wild	LLM Judge
ResearchRubrics	2025.11	Prompt-and-rubric-based deep research evaluation	LLM Judge / Human Judge
DEER	2025.12	Deep-research expert reports	LLM Judge
DeepSynth-Eval	2026.01	Information consolidation in deep survey writing	Rule-based
VideoDR-Benchmark	2026.01	Video deep research on open web	Rule-based
DeepResearch Bench II	2026.01	Rubric-based diagnosis of deep research reports	Rule-based
MMDeepResearch-Bench	2026.01	Multimodal deep research	Rule-based / LLM Judge
DeepSurvey-Bench	2026.01	Academic value of generated scientific surveys	Human Judge
DeepSearchQA	2026.01	Exhaustive answer-set generation	Rule-based / LLM Judge
deepsynth-bench	2026.02	Deep information synthesis	Rule-based / LLM Judge
		Software Engineering	
RepoBench	2023.06	Repository-level code auto-completion	Rule-based
SWE-bench	2023.10	Real-world GitHub issue resolution	Rule-based
CodeAgent	2024.01	Tool-integrated repo-level coding challenges	Rule-based
LiveCodeBench	2024.03	Holistic contamination-free code evaluation	Rule-based
DevEval	2024.05	Repository-aligned code generation	Rule-based
BigCodeBench	2024.06	Function calls and complex code generation	Rule-based
REPOCOD	2024.10	Realistic repository-level coding	Rule-based
HumanEval Pro /MBPP Pro	2024.12	Self-invoking code generation	Rule-based
Commit0	2024.12	Library generation from scratch	Rule-based
FEA-Bench	2025.03	Repository-level feature implementation	Rule-based
AutoCodeBench	2025.08	Automatic code benchmark generation	Rule-based
SWE-Bench Pro	2025.09	Long-horizon software engineering tasks	Rule-based
SWE-Sharp-Bench	2025.11	Reproducible C# software engineering tasks	Rule-based
LoCoBench-Agent	2025.11	Long-context software engineering	Rule-based
NL2Repo-Bench	2025.12	Long-horizon repository generation	Rule-based
OmniCode	2026.02	Diverse software engineering agent tasks	Rule-based
SWE-Universe	2026.02	Scalable verifiable software environments	Rule-based
FeatureBench	2026.02	Complex feature development	Rule-based
BeyondSWE	2026.03	Beyond single-repo bug fixing	Rule-based
		Tool Use and Function Calling	
API-Bank	2023.04	API planning and tool invocation	Rule-based
ToolLLM / ToolBench	2023.07	Large-scale real-world tool use	LLM Judge
AgentBench	2023.08	Multi-domain interactive agents	Rule-based
GAIA	2023.11	General AI assistant tasks	Rule-based
τ-bench	2024.06	Tool-agent-user interaction	Rule-based
AppWorld	2024.07	Controllable world of apps and people	Rule-based
AssistantBench	2024.07	Realistic web assistant tasks	Rule-based
ToolSandbox	2024.08	Stateful conversational tool use	Rule-based
ToolHop	2025.01	Query-driven multi-hop tool use	Rule-based
DICE-BENCH	2025.06	Multi-round, multi-party dialogue tool use	Rule-based
τ^2-Bench	2025.06	Conversational agents in dual-control settings	Rule-based
BFCL	2025.07	Function calling and agentic evaluation	Rule-based
FuncBenchGen	2025.09	Controllable multi-step function calling	Rule-based
GAIA2	2025.09	Real-life assistant tasks	Rule-based
CCTU	2026.03	Tool use under complex constraints	Rule-based
		Computer Use and GUI Grounding	
Mind2Web	2023.06	Generalist web agent interaction	Rule-based
WebArena	2023.07	Realistic web navigation	Rule-based
Android in the Wild	2023.07	Android device control	Rule-based
VisualWebArena	2024.01	Multimodal visual web tasks	Rule-based
WorkArena	2024.03	Common knowledge-work web tasks	Rule-based
OSWorld	2024.04	Open-ended real computer use	Rule-based
AndroidWorld	2024.05	Dynamic Android interaction environment	Rule-based
Windows Agent Arena	2024.09	Multi-modal desktop OS tasks	Rule-based
WorldGUI	2025.02	Dynamic testing for desktop GUI automation	Rule-based
ScreenSpot-Pro	2025.04	Professional high-resolution GUI grounding	Rule-based
MMBench-GUI	2025.07	Hierarchical multi-platform GUI evaluation	Rule-based
OSWorld-MCP	2025.10	MCP tool invocation in computer-use agents	Rule-based
VenusBench-GD	2025.12	Multi-platform GUI grounding	Rule-based
		ML Engineering and Scientific Research	
DS-1000	2022.11	Data science code generation	Rule-based
MLAgentBench	2023.10	Machine learning experimentation	Rule-based
DA-Code	2024.10	Agent data science code generation	Rule-based
ScienceAgentBench	2024.10	Data-driven scientific discovery	Rule-based
MLE-bench	2024.10	Machine learning engineering	Rule-based
PaperBench	2025.04	AI research replication	LLM Judge
TimeSeriesGym	2025.05	Time-series ML engineering	Rule-based
MLR-Bench	2025.05	Open-ended ML research	LLM Judge
FML-bench	2025.10	Automatic ML research	Rule-based
TRAJECT-Bench	2025.10	Trajectory-aware evaluation of agentic tool use	Rule-based / LLM Judge
ReplicatorBench	2026.02	Replicability in social and behavioral sciences	LLM Judge / Human Judge

5.2 Software Engineering

Software engineering benchmarks evaluate whether an agent can operate under realistic development conditions rather than merely generate standalone code. Evaluation has expanded from repository understanding and issue fixing to feature implementation and broader engineering workflows.

Repository Tasks. An early direction focuses on repository-grounded coding rather than isolated generation. These tasks ask whether an agent can work with project structure, cross-file dependencies, and documentation instead of relying only on a local prompt. RepoBench represents retrieval- and completion-oriented evaluation, while CodeAgentBench and REPOCOD move toward realistic repo-level generation with tool use and executable evaluation [401, 402, 56, 403]. Repository tasks primarily evaluate codebase grounding, context selection, and navigation, providing the foundation for later issue-resolution and engineering benchmarks.

Issue Resolution. A second line focuses on executable repair and issue-level debugging. SWE-bench makes real issue resolution the core task, SWE-Bench Pro extends this toward longer-horizon, multi-file, enterprise-grade problems, and LiveCodeBench adds executable, contamination-aware evaluation [404, 57, 60, 58, 405]. This direction emphasizes iterative correction, debugging, and responding to test feedback. As tasks grow longer-horizon, performance increasingly depends on whether the harness can retrieve context, run commands, inspect failures, and support repeated repair.

Broader Engineering Workflows. A third line broadens evaluation beyond issue repair to fuller engineering workflows. NL2Repo-Bench requires agents to design architecture, manage dependencies, and generate an installable repository from natural-language requirements [406]. FeatureBench focuses on feature-level implementation requiring coordinated edits across files [407]. BeyondSWE consolidates several beyond-bug-fixing regimes, including cross-repository reasoning, domain-specialized fixing, dependency migration, and doc-to-repo generation [408, 59]. These benchmarks collectively show that newer evaluations increasingly test global planning, cross-file consistency, and dependency management rather than local patch generation alone.

5.3 Tool Use and Function Calling

Tool use and function calling benchmarks evaluate whether an agent can invoke external capabilities correctly, relevantly, and consistently with task state. The overall trend moves from isolated function invocation toward multi-step, stateful, and conversational tool orchestration.

Function Calling. An early direction isolates the correctness of the call itself. API-Bank and BFCL evaluate whether a system can choose the appropriate function, supply valid arguments, and align the invocation with task requirements [409, 62, 393]. This targets a narrow but foundational layer of harness capacity: choosing and formatting the right action before longer trajectories begin.

Tool Use. A second line broadens scope from calling a tool to solving a task with tools. ToolBench, AgentBench, and GAIA evaluate whether agents can plan over APIs, integrate intermediate results, and use tools as part of broader task completion [115, 307, 410, 61]. These tasks emphasize tool routing, multi-step planning, and the ability to connect retrieved outputs to subsequent decisions, shifting evaluation from local call correctness toward coordinated tool-supported problem solving.

Stateful Interaction. More recent benchmarks move toward persistent orchestration across multiple turns and changing state. τ-bench and AssistantBench evaluate conversational interaction under user constraints, while AppWorld, ToolSandbox, and τ^2-Bench require managing shared app or tool state across longer trajectories [390, 132, 411, 64, 65, 63, 412]. These benchmarks place greater weight on state tracking, multi-turn consistency, and correct sequencing of dependent actions, though they still leave gaps around permissions, safety boundaries, partial-failure recovery, and cross-session memory.

5.4 Computer Use and GUI Grounding

Computer use and GUI grounding benchmarks evaluate whether an agent can perceive and act within interactive digital environments such as browsers, desktops, and mobile systems. Recent benchmarks increasingly stress multimodal perception, richer environments, and explicit GUI grounding.

Web Navigation. An early direction is browser-based task completion. Mind2Web and WebArena establish web navigation as a central evaluation setting by requiring systems to interpret page content, choose actions, and make progress across realistic sequences of web states [107, 108]. These tasks emphasize observation, action choice, and trajectory management inside an interface, introducing the browser as a meaningful environment for evaluating agentic interaction.

Computer Use. A second line expands from websites to broader digital environments. VisualWebArena adds visually grounded web interaction, while WorkArena, Android in the Wild, OSWorld, AndroidWorld, and Windows Agent Arena extend evaluation to enterprise platforms, mobile tasks, desktop systems, and real Windows control [104, 389, 413, 105, 69, 414, 67]. These benchmarks place more emphasis on multimodal perception, environment state, and broader computer control, moving evaluation closer to the conditions under which practical agents are deployed.

GUI Grounding. More recent work moves toward explicit grounding of actions in graphical interfaces. ScreenSpot-Pro and OSWorld-MCP make element identification and interface-aligned control a direct part of the task rather than an implicit subproblem [415, 70, 416, 68]. This reflects a shift toward general multimodal computer use, where precise grounding becomes a central bottleneck for reliable action. Even so, a substantial gap still remains between sandboxed settings and live deployment, and issues such as intervention points, safety-sensitive actions, latency, and the live deployment gap are still only weakly evaluated.

5.5 ML Engineering and Scientific Research

Benchmarks in this category evaluate whether an agent can support experimental, engineering, and scientific workflows over longer horizons than ordinary coding tasks. The development moves from data-science problem solving toward research replication and open-ended research assistance.

ML Experimentation. An early direction focuses on executable experimentation in data science and machine learning. DS-1000 evaluates realistic data-science coding, while MLAgentBench asks whether agents can iteratively run and improve experiments rather than merely generate code [417, 418, 394]. This setting emphasizes experiment setup, data processing, and empirical feedback, measuring whether the harness can support an experimental loop instead of a single generation step.

Scientific Workflows. Building on executable experimentation, a second line extends evaluation toward workflow-oriented scientific and ML-engineering tasks. DA-Code, ScienceAgentBench, and MLE-bench emphasize broader data-science workflows, scientific task decomposition, and ML-engineering pipelines over dependent steps [419, 10, 72, 71]. Performance increasingly depends on whether the harness can organize context, execution, and verification over extended workflows.

Research Replication. More recent benchmarks move toward replication and open-ended scientific validation. PaperBench, MLR-Bench, FML-bench, and ReplicatorBench evaluate whether agents can reproduce research results, pursue open-ended ML research, and reconstruct scientific claims under longer-horizon conditions [73, 420, 421, 422]. This represents a broader shift toward benchmarks that ask whether the system can support meaningful scientific work over extended trajectories. Scientific usefulness remains harder to verify than engineering correctness, with validity, novelty, computational inequality, and long-horizon reliability still open challenges.

Discussion

- Benchmark design is evolving from single-signal final-answer checking toward multi-signal evaluation: execution-based, state-based, rubric-based, and trajectory-aware signals together form a progression from verifying *what* was produced to evaluating *how* it was produced.

• Critical gaps remain: safety and permission control, operational costs (latency, token budgets), and long-term memory persistence are all weakly or inconsistently evaluated across current benchmarks.
• Most fundamentally, current benchmarks rarely disentangle improvements attributable to the base model from those attributable to the harness, limiting actionable insight for harness engineers.
• Closing these gaps—particularly cost-aware evaluation, safety testing, and model-harness attribution—will be central to the next generation of benchmarks.

6 Future Directions

The development of agentic harnesses is shifting from constructing capable execution loops to governing long-running runtime systems. Early harness designs focused on enabling models to perceive environments, invoke tools, maintain context, and coordinate actions. As these capabilities mature, the central bottleneck moves to what happens when agents run over extended horizons, interact with uncontrolled environments, persist state across sessions, and act with increasing authority. In such settings, performance depends not only on whether the model can produce a plausible next action but also on whether the harness can allocate compute selectively, manage state lifecycles, maintain environment-aligned beliefs, constrain risky actions, and expose evidence for evaluation.

This shift frames future harness engineering as runtime governance under resource, state, and action constraints. Efficiency concerns how compute, context, and tools are allocated under explicit budgets. Safety concerns how tool access, memory writes, and delegation are mediated before model outputs become real-world actions. Continual learning concerns how persistent memories and skills are consolidated, updated, and protected from drift. State and environment modeling concerns how partial observations and tool feedback are maintained as coherent belief states. Embodied harnesses test these abstractions under physical constraints where actions are noisy and often irreversible. Evaluation must then move beyond final success rates toward trace-aware, safety-aware, and cost-aware protocols. The following subsections discuss these directions in turn.

6.1 Efficiency

As LLM agents move from single-turn generation to long-horizon execution, efficiency becomes a harness-level concern rather than a model-level one alone. Cost in agentic systems accumulates through repeated model calls, expanding action-observation histories, tool-result feedback, retrieval, reflection, and multi-agent delegation. A uniform execution loop that applies the same model, context budget, and reasoning depth to every subtask is inherently wasteful, yet most current harnesses adopt exactly this uniform strategy. Model routing work offers a first alternative by framing inference as a cost-quality trade-off, showing that weaker and stronger models can be selected dynamically without always invoking the most capable model [423]. The same principle extends to reasoning depth: analyses of chain-of-thought prompting caution that deeper reasoning traces are not universally beneficial, since additional reasoning can be brittle, task-specific, and costly when it does not generalize [424]. Taken together, these results suggest that agent efficiency should be studied as a scheduling problem: the harness must decide when deeper reasoning, stronger models, or retrieval are warranted, and when cheaper execution paths suffice.

While model selection and reasoning depth determine the cost of each individual step, a second bottleneck arises from what accumulates across steps: context and trajectory management. Long-running agents repeatedly reprocess growing histories, observations, tool outputs, and intermediate decisions. ACON addresses this by optimizing compression guidelines against downstream agent failures, showing that context compression must preserve task-relevant information rather than simply shorten text [277]. AgentDiet complements this finding from the removal side, demonstrating that long-horizon trajectories contain redundant and expired information that can be pruned at inference time without sacrificing task performance [425]. More recent frameworks move beyond such passive compression toward active context governance. ARC treats context as a dynamic reasoning state that can be monitored and revised when context rot appears, while Context as a Tool makes context maintenance a callable action inside long-horizon SWE agents [426, 427]. Together, these works

indicate that efficient harnesses should treat history not as an immutable transcript but as a managed resource whose fidelity, lifetime, and accessibility are governed by the runtime.

Beyond what the model computes and what context it sees, efficiency also depends on how tools and reasoning components are orchestrated. Tool use can reduce model burden by delegating computation to external systems, but poorly structured invocation introduces additional calls, long feedback traces, and expensive replanning that offset the delegation benefit. Recent work on layered tool orchestration shows that separating global dependency guidance from local reflective correction can reduce this overhead, avoiding full-trajectory replanning after each tool-level failure [428]. These three dimensions form a coherent efficiency stack: routing controls which model is used, context management controls what the model sees, trajectory reduction controls what history persists, and tool orchestration controls how much reasoning is spent coordinating external actions. Future harnesses should therefore treat computation as a budgeted runtime resource, allocating capacity where uncertainty or task risk justifies it while collapsing routine subtasks into cheaper model calls, compressed state, cached artifacts, or structured tool execution.

6.2 Safety

Safety in agentic systems is shifting from output alignment to runtime capability governance. When an agent only produces text, safety can be framed around harmful responses or preference alignment. Once the harness allows a model to browse untrusted content, call tools, write to external systems, store persistent memory, or delegate subtasks, the primary risk surface moves from generation into the observation-to-action loop. This expanded attack surface manifests at progressively deeper layers of the harness. At the tool-output level, AgentDojo showed that returned results can carry prompt-injection attacks that steer agents toward malicious actions [429]. At the tool-metadata level, MCPTox demonstrated that MCP-style ecosystems introduce poisoning risks before a tool is even executed [430]. At the memory level, AgentSys argues that indirect prompt injection becomes persistent when untrusted observations accumulate in working memory, and proposes hierarchical memory isolation as a defense [32]. This progression indicates that harness safety concerns not only what the model generates, but what information, tools, and authority the runtime allows to influence future actions.

Recognizing that the threat surface spans the entire runtime, recent work increasingly treats safety as a governance architecture problem. At the conceptual level, OpenPort Protocol frames tool access as a governance interface combining authorization-dependent discovery, scoped permissions, stable failure semantics, risk-gated execution, state revalidation, and structured audit events [431]. At the formal level, Runtime Governance for AI Agents shows that many safety policies are path-dependent, meaning that whether an action is acceptable depends on the agent identity, partial execution trace, and previous data accesses, not on the next action in isolation [432]. At the enforcement level, AgentSpec provides a domain-specific language for specifying triggers, predicates, and interventions over agent actions, while AARM proposes an action-interception layer that accumulates session context, checks policy alignment, and records tamper-evident receipts [433, 434]. SafeHarness integrates these concerns across the full agent lifecycle, combining adversarial context filtering, causal verification, privilege-separated tool control, and rollback with adaptive degradation [435]. Together, these works signal a shift from prompt-level safeguards toward lifecycle-aware mediation inside the harness.

These governance mechanisms, however, largely address individual actions or single-layer threats. The central open problem is compositional governance over long horizons, where unsafe behavior emerges not from any single impermissible step but from combinations of individually permissible ones: reading sensitive data before sending an email, invoking a benign tool after poisoned metadata, retrieving stale memory under a changed policy, or delegating partial context without preserving provenance. Addressing such compositional risks requires safety mechanisms that travel with state. Tool outputs, memory entries, retrieved documents, and planned actions should carry provenance, trust, and freshness metadata checkable before downstream use. High-risk actions should support approval gates, state revalidation, rollback when possible, and auditable receipts when execution cannot be undone. This formulation clarifies how harness-level safety differs from model-level alignment: alignment shapes behavioral tendencies, whereas harness governance determines which capabilities are exposed, which information persists, which actions may execute, and what evidence must be produced before and after execution.

6.3 Continual Learning

Continual learning in agentic systems should not be equated with continually updating model weights. Agents improve through multiple layers of adaptation: the model may be fine-tuned, the harness may update its tools, rules, and policies, and the context layer may accumulate memories, user preferences, and reusable skills. Recent product-side discussions make this layered view explicit by separating model-level, harness-level, and context-level learning. A useful theoretical grounding is the externalization view of LLM agents, which observes that memory externalizes state across time, skills externalize procedural expertise, protocols externalize interaction structure, and the harness coordinates these externalized components at runtime. Combining these perspectives, continual learning becomes less about whether an agent has a larger memory store, and more about how the harness maintains the external structures through which past experience influences future behavior [436].

If continual learning operates primarily over externalized state, the core challenge shifts from memory accumulation to context lifecycle management. Early systems such as MemGPT introduced structured memory tiers rather than treating the context window as an append-only transcript [148]. Subsequent work made memory operations themselves learnable: A-MEM differentiates memory construction and retrieval strategies, while Memory as Action treats memory editing as part of the agent policy rather than a fixed heuristic [31, 437]. Production deployments further validate this trend by operationalizing distinct lifecycle stages. Anthropic's context-management guidance, for instance, separates compaction, tool-result clearing, and memory: compaction compresses the current conversation, tool-result clearing removes re-fetchable outputs, and memory persists structured notes across sessions [438]. The underlying insight is that different types of state require different lifecycle treatments: some should remain in active context, some should be summarized, some can be dropped because it is reconstructable, and some must persist outside the window for future sessions. Continual learning thus becomes a governed write-manage-read loop over states, not a simple retrieval problem.

The same lifecycle challenge becomes even more acute when it extends from declarative memories to procedural skills. Agent Workflow Memory shows that agents can induce reusable workflows from prior trajectories and retrieve them to guide future action sequences [146]. More recent skill-oriented systems treat such procedures as explicit packages: skills may include instructions, scripts, metadata, tests, and execution constraints, and can be dynamically discovered, loaded, shared, or versioned by the harness. Skills, however, are more tightly coupled to external dependencies than declarative memories, and consequently acquire software-like maintenance risks once they become durable objects. Summaries can overcompress rare exceptions, old beliefs can survive after environment changes, tool APIs can invalidate procedural routines, and low-quality memories can contaminate future behavior. Future work should therefore study continual learning as governed persistence: how to consolidate experience, preserve provenance, expire stale state, reconcile contradictory updates, version skills, and keep persistent structures aligned with the tools, environments, and users.

6.4 State and Environment Modeling

State and environment modeling in agentic harnesses is not primarily about whether an LLM contains latent commonsense knowledge. The more immediate question is whether the runtime can maintain a compact, updatable, and action-relevant representation of the environment over long trajectories. For practical agents, the external world is only partially exposed through observations such as web page changes, GUI layouts, repository files, tool responses, sandbox feedback, and intermediate artifacts. Because these observations are incomplete and transient, treating the full interaction history as a proxy for state is both inefficient and brittle: the history grows with redundant or expired information, while the model's implicit belief may silently drift from the actual environment. A harness therefore needs explicit mechanisms for constructing and updating a belief state that supports planning, execution, and recovery, rather than relying on raw observation replay.

Recent research addresses this need from two complementary directions that correspond to two distinct planning requirements. The first direction focuses on explicit belief maintenance, which answers the question of where the agent currently stands. WorldCoder learns executable environment models by writing code and interacting with the environment, enabling planning over learned models rather than raw histories [439]. CoEx couples planning with a persistent neurosymbolic belief state updated from experience, while PABU shows that progress-aware belief updates can outperform full-history conditioning by retaining only decision-relevant state [440]. The second direction models

action-conditioned transitions, which answers the question of what will happen if the agent acts. Reinforcement World Model Learning trains agents to predict next states after actions and uses discrepancies between simulated and real feedback as learning signals, making state transition itself part of harness optimization. Related work in web, GUI, and mobile environments similarly uses state machines or transition graphs to represent interface dynamics and action consequences. Together, these two capabilities enable an agent to reason about both its current situation and the likely outcomes of candidate actions, which are the minimal requirements for informed planning.

While academic research pursues learned state models, deployed systems often achieve the same function through structured infrastructure. Coding and computer-use agents maintain environment state through repository files, terminal outputs, test results, execution logs, and sandbox permissions. Recent product-side runtimes make this explicit: controlled sandbox environments define what the agent can inspect, execute, and modify, while progressive disclosure determines which parts are surfaced at each step. Agent-World extends this view by arguing that robust agent development requires scalable environment synthesis, sandboxing, and realistic state transitions [441]. Whether realized through learned models or engineered infrastructure, the underlying challenge is the same: maintaining consistency between what the agent believes and what is actually true. Future work should therefore treat state modeling as a harness-level consistency problem: how to reconcile model priors, partial observations, tool feedback, memory, and changing external state; how to detect stale or uncertain beliefs; and how to expose reusable state-transition interfaces across web, code, GUI, and embodied settings.

6.5 Embodied Harnesses

Embodied settings provide a particularly demanding test for harness engineering because state is partially observable, execution is noisy, feedback is real-time, and actions are often irreversible. Earlier systems such as PaLM-E and RT-2 showed that multimodal models can connect visual and linguistic knowledge to embodied planning [314, 315], but these systems largely treated the model as a monolithic planner-to-actor pipeline. Recent work decomposes this pipeline into explicit harness layers. Gemini Robotics-ER 1.6 separates high-level embodied reasoning, including spatial reasoning, task planning, and success detection, from low-level VLA execution and user-defined functions. Green-VLA similarly frames generalist robot deployment as a staged control stack with task planning, progress prediction, out-of-distribution detection, and RL-based policy alignment above a unified action interface [442]. ACoT-VLA and ST4VLA reinforce this decomposition by making action-space reasoning and spatial grounding learning objectives within VLA training [443, 444]. The converging lesson is that embodied progress depends less on end-to-end action generation and more on an explicit runtime boundary between high-level semantic reasoning and low-level motor execution.

Once this boundary is established, grounded state management becomes the harness's core responsibility at the semantic level. Unlike digital environments, the physical world does not expose complete or rollback-friendly state, so the harness must maintain an approximate belief state from camera streams, sensor readings, and execution feedback. Three runtime questions then arise continuously: whether the current observation is reliable enough to plan from, whether a candidate action remains valid given updated perception, and whether the task has progressed as expected or drifted out of distribution. Recent systems address these questions through concrete mechanisms: Gemini Robotics-ER 1.6 provides multi-view success detection and spatial pointing for progress assessment; Green-VLA adds real-time progress tracking and OOD monitoring for drift detection [442]; and RoboAgent decomposes embodied task planning into scheduler-mediated capabilities with separate contexts for reliability isolation [445]. The common architectural pattern is that embodied harnesses must connect perception, planning, belief update, and success detection into a continuous runtime loop, since treating perception as a one-shot input cannot sustain multi-step physical execution.

The irreversibility of physical actions further demands pre-execution validation and recovery protocols that go beyond what software agents require. In digital environments, failed actions can often be retried or sandboxed; in physical environments, a single unsafe grasp or collision may damage the scene or the robot itself. EmbodiedBench confirms that even strong multimodal models struggle with sustained grounding and robust action selection under these conditions [5]. The runtime therefore cannot rely on model capability alone but must verify that planned actions are compatible with current perception, physical constraints, and safety rules before dispatching them to the controller. AgentSpec provides one useful abstraction by treating such constraints as executable harness logic

rather than informal prompts [446]. Production robotics workflows operationalize the same principle at scale: NVIDIA's simulation-to-production stack and Isaac Lab-Arena require simulation evaluation, software-in-the-loop testing, digital-twin fleet testing, and safety guardrails before real-world execution. When validation passes but execution still fails, the harness must then support re-observation, replanning, controller switching, safe-state retreat, and escalation to human supervision. In this sense, embodied harnesses serve less as a separate application domain than as a demanding testbed for whether the planning, state maintenance, validation, and recovery abstractions developed for digital agents can survive contact with physical reality.

6.6 Evaluation

Evaluation for agentic harnesses is shifting from measuring isolated task success to assessing the quality of complete execution runs. Recent benchmarks already move beyond static question answering by placing agents in longer, more interactive settings: BrowseComp and OSWorld stress multi-step browsing and computer-use trajectories, PaperBench and ReplicatorBench evaluate open-ended research workflows, and BeyondSWE targets software-engineering tasks whose failures emerge only after cross-repository reasoning [313, 105, 73, 422, 408]. These benchmarks extend task horizon and difficulty, yet they still permit evaluation based primarily on the final output. More recent benchmarks go further by making the environment itself dynamic, which forces evaluation to consider the entire execution trajectory. Gaia2 evaluates agents in asynchronous environments where state changes independently of agent actions, while Claw-Eval-Live separates refreshable workflow signals from reproducible snapshots and grades runs using execution traces, audit logs, and workspace artifacts [66, 447]. Once the environment can change mid-execution, a correct final answer no longer guarantees a sound process, making the agent run itself the necessary unit of measurement.

Treating the run as the evaluation object, however, immediately raises the question of what within that run should be measured. Final success rates hide whether an agent followed a dependable procedure or arrived at a correct end state by chance, excessive retries, or unobserved side effects. This attribution problem motivates a second evaluation requirement: process visibility. Existing benchmarks have begun exposing richer intermediate signals: ResearchRubrics evaluates long-form research through structured rubrics, OSWorld-MCP makes tool invocation patterns explicit, and AgentLongBench evaluates long-context agents through environment rollouts rather than passive retrieval [400, 70, 448]. These efforts point toward a broader standard: benchmarks should record how state is updated, which tools are called, what intermediate artifacts are produced, how failures are handled, and whether memory becomes stale over time. Without such trace-level evidence, it remains difficult to attribute performance to the model, the harness, the tools, or the surrounding execution environment.

Process visibility reveals what an agent did, but a complete evaluation must also judge whether what it did was safe and affordable. This motivates a third requirement: evaluating governance, safety, and cost as first-class metrics rather than secondary diagnostics. Agentic failures increasingly occur inside trajectories where they are invisible to output-only evaluation: unsafe tool calls, poisoned tools, contaminated memory, privacy leakage through internal messages, and missing approval steps may never surface in the final response. ATBench makes this trajectory-level safety problem explicit through heterogeneous tool pools and delayed-trigger risks, while AgentLeak demonstrates that output-only audits miss privacy leakage through inter-agent messages and shared memory [449, 450]. Cost exhibits a parallel evaluation gap. General AgentBench shows that test-time scaling is limited by context ceilings and verification gaps, implying that pass rates should be interpreted together with token use, latency, tool-call count, and verification budget [451]. Future harness benchmarks should therefore report path quality, state quality, safety quality, and cost quality as an integrated profile, distinguishing systems that merely solve benchmark instances from harnesses that are reliable, auditable, and deployable.

7 Conclusion

This paper presented a systematic review of agent systems from the perspective of harness engineering. Starting from the observation that practical capability depends as much on system scaffolding as on the underlying model, we formalized the harness as the orchestration layer governing execution flow, tool invocation, memory management, context construction, and safety enforcement, and framed

harness engineering as the joint optimization of both the scaffold and the model it supports. Under this framework, we examined how agent workflows, memory systems, skill libraries, and multi-agent orchestration collectively determine system-level performance, and reviewed optimization strategies spanning context engineering and agentic training, as well as evaluation benchmarks across software engineering, deep research, tool use, computer use, and scientific discovery.

This perspective shifted the analytical focus from what agents can accomplish to how the surrounding infrastructure enables reliable accomplishment. Our analysis revealed that many practical failures in deployed systems, including instability during long-horizon execution, inefficient tool utilization, context degradation, and suboptimal multi-agent coordination, frequently originated from inadequate harness design rather than model-level deficiency alone. These failures reflected a structural mismatch between the single-turn generative interface of LLMs and the stateful, iterative nature of real-world problem solving, reinforcing the central thesis that the relationship between model and harness is inherently synergistic: improvements to one unlock latent potential in the other.

Looking ahead, several directions merit investigation: improving the scalability and robustness of harness execution in long-horizon settings, developing principled methods for joint scaffold-model optimization, generalizing agent harnesses to physical and embodied domains, and establishing evaluation protocols that disentangle harness contributions from model capability. We hope this paper serves as a practical reference for building more reliable, scalable, and controllable agent systems through principled harness engineering.

References

[1] Michael J. Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *Knowl. Eng. Rev.*, 10(2):115–152, 1995.

[2] Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025.

[3] Ho Chit Siu, Jaime Daniel Peña, Edenna Chen, Yutai Zhou, Victor J. Lopez, Kyle Palko, Kimberlee C. Chang, and Ross E. Allen. Evaluation of human-ai teams for learned and rule-based agents in hanabi. In *NeurIPS*, pages 16183–16195, 2021.

[4] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

[5] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[6] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

[7] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[8] Huatong Song, Lisheng Huang, Shuang Sun, Jinhao Jiang, Ran Le, Daixuan Cheng, Guoxin Chen, Yiwen Hu, Zongchao Chen, Wayne Xin Zhao, et al. Swe-master: Unleashing the potential of software engineering agents via post-training. *arXiv preprint arXiv:2602.03411*, 2026.

[9] Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. A survey on code generation with llm-based agents. *CoRR*, abs/2508.00083, 2025.

[10] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *ICLR*. OpenReview.net, 2025.

[11] OpenClaw. Openclaw, 2026.

[12] Hermes. Hermes agent, 2026.

[13] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *CoRR*, abs/2503.21460, 2025.

[14] Zhipeng Liu, Xuefeng Bai, Kehai Chen, Xinyang Chen, Xiucheng Li, Yang Xiang, Jin Liu, Hong-Dong Li, Yaowei Wang, Liqiang Nie, and Min Zhang. A survey on the feedback mechanism of llm-based AI agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10582–10592. ijcai.org, 2025.

[15] Aske Plaat, Max J. van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. Agentic large language models, a survey. *J. Artif. Intell. Res.*, 84, 2025.

[16] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *ICLR*. OpenReview.net, 2023.

[17] Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. A survey of context engineering for large language models. *CoRR*, abs/2507.13334, 2025.

[18] Apoorva Adimulam, Rajesh Gupta, and Sumit Kumar. The orchestration of multi-agent systems: Architectures, protocols, and enterprise adoption. *CoRR*, abs/2601.13671, 2026.

[19] Mohamad Abou Ali, Fadi Dornaika, and Jinan Charafeddine. Agentic ai: a comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, 59(1):11, 2025.

[20] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of LLM agents: A survey. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy, editors, *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pages 6129–6139. ACM, 2025.

[21] Ling Yue, Kushal Raj Bhandari, Ching-Yun Ko, Dhaval Patel, Shuxin Lin, Nianjun Zhou, Jianxi Gao, Pin-Yu Chen, and Shaowu Pan. From static templates to dynamic runtime graphs: A survey of workflow optimization for llm agents, 2026.

[22] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 22–30. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

[23] Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. GAP: graph-based agent planning with parallel tool use and reinforcement learning. *CoRR*, abs/2510.25320, 2025.

[24] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[25] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

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

[27] Gunshi Gupta, Karmesh Yadav, Zsolt Kira, Yarin Gal, and Rahaf Aljundi. Memo: Training memory-efficient embodied agents with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. Spotlight Presentation.

[28] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics.

[29] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI*, Frontiers in Artificial Intelligence and Applications, pages 2993–3000. IOS Press, 2025.

[30] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, Mar. 2024.

[31] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, 2025.

[32] Ruoyao Wen, Hao Li, Chaowei Xiao, and Ning Zhang. Agentsys: Secure and dynamic LLM agents through explicit hierarchical memory management. *CoRR*, abs/2602.07398, 2026.

[33] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *CoRR*, abs/2602.08234, 2026.

[34] Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

[35] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills - beyond tool use in LLM agents. *CoRR*, abs/2602.20867, 2026.

[36] Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. *CoRR*, abs/2508.04700, 2025.

[37] Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[38] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling (COLM 2024)*, 2024.

[39] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[40] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[41] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *CoRR*, abs/2308.10848, 2023.

[42] Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *CoRR*, abs/2506.12508, 2025.

[43] Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. In *ICLR*. OpenReview.net, 2025.

[44] Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models. *arXiv preprint arXiv:2502.01142*, 2025.

[45] Xintao Wang, Heng Wang, Yifei Zhang, Xinfeng Yuan, Rui Xu, Jen-tse Huang, Siyu Yuan, Haoran Guo, Jiangjie Chen, Shuchang Zhou, Wei Wang, and Yanghua Xiao. Coser: Coordinating llm-based persona simulation of established roles. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[46] Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Improving the efficiency of LLM agent systems through trajectory reduction. *CoRR*, abs/2509.23586, 2025.

[47] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *EuroSys*, pages 1279–1297. ACM, 2025.

[48] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025.

[49] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025.

[50] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, 2025.

[51] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *CoRR*, abs/2504.13958, 2025.

[52] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *CoRR*, abs/2504.12516, 2025.

[53] Abhinav Java, Ashmit Khandelwal, Sukruta Prakash Midigeshi, Aaron Halfaker, Amit Deshpande, Navin Goyal, Ankur Gupta, Nagarajan Natarajan, and Amit Sharma. Characterizing deep research: A benchmark and formal definition. *CoRR*, abs/2508.04183, 2025.

[54] Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. Widesearch: Benchmarking agentic broad info-seeking. *CoRR*, abs/2508.07999, 2025.

[55] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. *CoRR*, abs/2506.11763, 2025.

[56] Shanchao Liang, Nan Jiang, Yiran Hu, and Lin Tan. Can language models replace programmers for coding? REPOCOD says 'not yet'. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 24698–24717. Association for Computational Linguistics, 2025.

[57] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can AI agents solve long-horizon software engineering tasks? *CoRR*, abs/2509.16941, 2025.

[58] Lilin Wang, Lucas Ramalho, Alan Celestino, Phuc Anthony Pham, Yu Liu, Umang Kumar Sinha, Andres Portillo, Onassis Osunwa, and Gabriel Maduekwe. Swe-bench++: A framework for the scalable generation of software engineering benchmarks from open-source repositories. *CoRR*, abs/2512.17419, 2025.

[59] Atharv Sonwane, Eng-Shen Tu, Wei-Chung Lu, Claas Beger, Carter Larsen, Debjit Dhar, Simon Alford, Rachel Chen, Ronit Pattanayak, Tuan Anh Dang, Guohao Chen, Gloria Geng, Kevin Ellis, and Saikat Dutta. Omnicode: A benchmark for evaluating software engineering agents. *CoRR*, abs/2602.02262, 2026.

[60] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[61] Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen.

Toolhop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 2995–3021. Association for Computational Linguistics, 2025.

[62] Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): from tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[63] Kyochul Jang, Donghyeon Lee, Kyusik Kim, Dongseok Heo, Taewhoo Lee, Woojeong Kim, and Bongwon Suh. DICE-BENCH: evaluating the tool-use capabilities of large language models in multi-round, multi-party dialogues. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 26822–26846. Association for Computational Linguistics, 2025.

[64] Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. 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, 2025.

[65] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *CoRR*, abs/2506.07982, 2025.

[66] Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025.

[67] Henry Hengyuan Zhao, Difei Gao, and Mike Zheng Shou. Worldgui: Dynamic testing for comprehensive desktop GUI automation. *CoRR*, abs/2502.08047, 2025.

[68] Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yuan Huang, Yiqian Liu, Xiao Zhang, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. Mmbench-gui: Hierarchical multi-platform evaluation framework for GUI agents. *CoRR*, abs/2507.19478, 2025.

[69] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[70] Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. Osworld-mcp: Benchmarking mcp tool invocation in computer-use agents. *arXiv preprint arXiv:2510.24563*, 2025.

[71] Yujie Liu, Zonglin Yang, Tong Xie, Jinjie Ni, Ben Gao, Yuqiang Li, Shixiang Tang, Wanli Ouyang, Erik Cambria, and Dongzhan Zhou. Researchbench: Benchmarking llms in scientific discovery via inspiration-based task decomposition. *CoRR*, abs/2503.21248, 2025.

[72] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[73] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate AI research. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan

Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[74] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[75] Christopher Cruz. Ai runtime infrastructure. *arXiv preprint arXiv:2603.00495*, 2026.

[76] Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang H. Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The openhands software agent SDK: A composable and extensible foundation for production agents. *CoRR*, abs/2511.03690, 2025.

[77] Naveen Kumar Krishnan. Beyond context sharing: A unified agent communication protocol (acp) for secure, federated, and autonomous agent-to-agent (a2a) orchestration. *arXiv preprint arXiv:2602.15055*, 2026.

[78] Hao Ke. Quine: Realizing llm agents as native posix processes. *arXiv preprint arXiv:2603.18030*, 2026.

[79] Luoxi Meng, Henry Feng, Ilia Shumailov, and Earlence Fernandes. cellmate: Sandboxing browser AI agents. *CoRR*, abs/2512.12594, 2025.

[80] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 50208–50232. PMLR / OpenReview.net, 2024.

[81] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. Deepagent: A general reasoning agent with scalable toolsets. *CoRR*, abs/2510.21618, 2025.

[82] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[83] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: a survey. *Frontiers Comput. Sci.*, 19(8):198343, 2025.

[84] Kai Mei, Xi Zhu, Wujiang Xu, Mingyu Jin, Wenyue Hua, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. Aios: Llm agent operating system. In *Second Conference on Language Modeling*.

[85] Ruixin Zhang, Wuyang Dai, Hung Viet Pham, Gias Uddin, Jinqiu Yang, and Song Wang. Engineering pitfalls in ai coding tools: An empirical study of bugs in claude code, codex, and gemini cli, 2026.

[86] Nghi D. Q. Bui. Building effective AI coding agents for the terminal: Scaffolding, harness, context engineering, and lessons learned. *CoRR*, abs/2603.05344, 2026.

[87] Varun Ursekar, Apaar Shanker, Veronica Chatrath, Sam Denton, et al. Vero: An evaluation harness for agents to optimize agents. *arXiv preprint arXiv:2602.22480*, 2026.

[88] Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2026. Accessed 2026-03-27.

[89] Yue Xu, Qian Chen, Zizhan Ma, Dongrui Liu, Wenxuan Wang, Xiting Wang, Li Xiong, and Wenjie Wang. Toward personalized llm-powered agents: Foundations, evaluation, and future directions. *arXiv preprint arXiv:2602.22680*, 2026.

[90] Rebecca Westhäußer, Wolfgang Minker, and Sebastian Zepf. Enabling personalized long-term interactions in llm-based agents through persistent memory and user profiles. *CoRR*, abs/2510.07925, 2025.

[91] Yuren Hao, Shuhaib Mehri, ChengXiang Zhai, and Dilek Hakkani-Tür. User preference modeling for conversational llm agents: Weak rewards from retrieval-augmented interaction. *arXiv preprint arXiv:2603.20939*, 2026.

[92] Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to AI memory: A survey on memory mechanisms in the era of llms. *CoRR*, abs/2504.15965, 2025.

[93] Hyunsung Cho, Jacqui Fashimpaur, Naveen Sendhilnathan, Jonathan Browder, David Lindlbauer, Tanya R. Jonker, and Kashyap Todi. Persistent assistant: Seamless everyday AI interactions via intent grounding and multimodal feedback. In Naomi Yamashita, Vanessa Evers, Koji Yatani, Sharon Xianghua Ding, Bongshin Lee, Marshini Chetty, and Phoebe O. Toups Dugas, editors, *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI 2025, YokohamaJapan, 26 April 2025- 1 May 2025*, pages 59:1–59:19. ACM, 2025.

[94] Peter Steinberger. Introducing openclaw. https://openclaw.ai/blog/introducing-openclaw, 2026. OpenClaw official blog, accessed 2026-03-27.

[95] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[96] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2609–2634. Association for Computational Linguistics, 2023.

[97] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering for large language models. *Patterns*, 6(7):101260, 2025.

[98] Jinyang Chen, Haolun Wu, Jianhong Pang, Yihua Wang, Dell Zhang, and Changzhi Sun. Tool learning with language models: a comprehensive survey of methods, pipelines, and benchmarks. *Vicinagearth*, 2(1):16, 2025.

[99] Haoyuan Xu, Chang Li, Xinyan Ma, Xianhao Ou, Zihan Zhang, Tao He, Xiangyu Liu, Zixiang Wang, Jiafeng Liang, Zheng Chu, Runxuan Liu, Rongchuan Mu, Ming Liu, and Bing Qin. The evolution of tool use in llm agents: From single-tool call to multi-tool orchestration, 2026.

[100] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[101] Xinbin Liang, Jinyu Xiang, Zhaoyang Yu, Jiayi Zhang, Sirui Hong, Sheng Fan, Xiao Tang, Bang Liu, Yuyu Luo, and Chenglin Wu. Openmanus: An open-source framework for building general ai agents, 2025. GitHub repository.

[102] OpenAI. Unrolling the codex agent loop, 2026. Developer documentation.

[103] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[104] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins,

and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 881–905. Association for Computational Linguistics, 2024.

[105] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[106] Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: an open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[107] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[108] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[109] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for llm-based web agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[110] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 61349–61385. PMLR, 21–27 Jul 2024.

[111] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[112] Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[113] Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. In *NeurIPS*, 2024.

[114] Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. Gui-actor: Coordinate-free visual grounding for gui agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025.

[115] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*.

[116] Zichao Li, Yanshuai Cao, and Jackie CK Cheung. Do LLMs build world representations? probing through the lens of state abstraction. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[117] Hayeong Lee, Jun Ho Seo, Sunguk Shin, Jinho Lee, Myunsoo Kim, Minsuk Chang, and Byung-Jun Lee. Evaluating LLM planning in partially observable environments via observation representations and action sequences. In *NeurIPS 2025 Workshop on Bridging Language, Agent, and World Models (LAW)*, 2025.

[118] Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. *CoRR*, abs/2508.21433, 2025.

[119] Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. In *ICLR*. OpenReview.net, 2025.

[120] Hang Ding, Peidong Liu, Junqiao Wang, Ziwei Ji, Meng Cao, Rongzhao Zhang, Lynn Ai, Eric Yang, Tianyu Shi, and Lei Yu. Dynaweb: Model-based reinforcement learning of web agents. *CoRR*, abs/2601.22149, 2026.

[121] Yi Fang, Bowen Jin, Jiacheng Shen, Sirui Ding, Qiaoyu Tan, and Jiawei Han. Graphgpt-o: Synergistic multimodal comprehension and generation on graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19467–19476, June 2025.

[122] Jiani Huang, Mayank Keoliya, Matthew Kuo, Neelay Velingker, Amish Sethi, JungHo Jung, Ziyang Li, Ser Nam Lim, and Mayur Naik. Esca: Contextualizing embodied agents via scene-graph generation. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

[123] Anthropic. Building a c compiler with a team of parallel claudes, 2026. Blog post.

[124] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[125] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

[126] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, pages 17682–17690. AAAI Press, 2024.

[127] Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing LLM reasoning with agentic tools. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28489–28503, Vienna, Austria, July 2025. Association for Computational Linguistics.

[128] Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. Agentic reasoning for large language models. *CoRR*, abs/2601.12538, 2026.

[129] Anthropic. Effective harnesses for long-running agents, 2025. Blog post.

[130] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Y. Zou. Avatar: Optimizing

LLM agents for tool usage via contrastive reasoning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[131] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023.

[132] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL (1)*, pages 16022–16076. Association for Computational Linguistics, 2024.

[133] Zhiwei Zhang, Fei Zhao, Rui Wang, Zezhong Wang, Bin Liang, Jiakang Wang, Yao Hu, Shaosheng Cao, and Kam-Fai Wong. Robust tool use via fission-grpo: Learning to recover from execution errors, 2026.

[134] Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning, 2025.

[135] Meng Lu, Ran Xu, Yi Fang, Wenxuan Zhang, Yue Yu, Gaurav Srivastava, Yuchen Zhuang, Mohamed Elhoseiny, Charles Fleming, Carl Yang, Zhengzhong Tu, Yang Xie, Guanghua Xiao, Hanrui Wang, Di Jin, Wenqi Shi, and Xuan Wang. Scaling agentic reinforcement learning for tool-integrated reasoning in vlms, 2025.

[136] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*, 2025.

[137] Yihan Wang, Peiyu Liu, Runyu Chen, and Wei Xu. Beyond static pipelines: Learning dynamic workflows for text-to-sql. *CoRR*, abs/2602.15564, 2026.

[138] Yihan Wang, Peiyu Liu, Runyu Chen, Jiaxing Pu, and Wei Xu. Squrve: A unified and modular framework for complex real-world text-to-sql tasks. *CoRR*, abs/2510.24102, 2025.

[139] Anthropic. Code execution with mcp: building more efficient ai agents, 2025. Blog post.

[140] Andrew Szot, Bogdan Mazoure, Harsh Agrawal, R. Devon Hjelm, Zsolt Kira, and Alexander Toshev. Grounding multimodal large language models in actions. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[141] Sai Puppala, Ismail Hossain, Md Jahangir Alam, Yoonpyo Lee, Jay Yoo, Tanzim Ahad, Syed Bahauddin Alam, and Sajedul Talukder. Agent-fence: Mapping security vulnerabilities across deep research agents, 2026.

[142] Anthropic. Managing context on the claude developer platform, 2025. Blog post.

[143] Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning*, pages 26396–26415. PMLR, 2024.

[144] Islem Bouzenia and Michael Pradel. Understanding software engineering agents: A study of thought-action-result trajectories. In *40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025, Seoul, Korea, Republic of, November 16-20, 2025*, pages 2846–2857. IEEE, 2025.

[145] Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. Graph-based agent memory: Taxonomy, techniques, and applications. *CoRR*, abs/2602.05665, 2026.

[146] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[147] Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[148] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023.

[149] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[150] Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

[151] Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. Hello again! LLM-powered personalized agent for long-term dialogue. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5259–5276, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[152] Yaxiong Wu, Yongyue Zhang, Sheng Liang, and Yong Liu. Sgmem: Sentence graph memory for long-term conversational agents. *CoRR*, abs/2509.21212, 2025.

[153] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *CoRR*, abs/2501.13956, 2025.

[154] Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *CoRR*, abs/2601.01885, 2026.

[155] Yiweng Xie, Bo He, Junke Wang, Xiangyu Zheng, Ziyi Ye, and Zuxuan Wu. Fluxmem: Adaptive hierarchical memory for streaming video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[156] LangChain. Memory overview, 2026. Documentation.

[157] LangChain. LangMem, 2026. Documentation.

[158] Young Bin Park. Graph-native cognitive memory for ai agents: Formal belief revision semantics for versioned memory architectures. *arXiv preprint arXiv:2603.17244*, 2026.

[159] Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche, editors, *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM, 2023.

[160] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913, 2024.

[161] Yujie Zhang, Weikang Yuan, and Zhuoren Jiang. Bridging intuitive associations and deliberate recall: Empowering LLM personal assistant with graph-structured long-term memory. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17533–17547, Vienna, Austria, July 2025. Association for Computational Linguistics.

[162] Johannes Kirmayr, Lukas Stappen, Phillip Schneider, Florian Matthes, and Elisabeth Andre. CarMem: Enhancing long-term memory in LLM voice assistants through category-bounding. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, Steven Schockaert, Kareem Darwish, and Apoorv Agarwal, editors, *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 343–357, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[163] Weishu Chen, Jinyi Tang, Zhouhui Hou, Shihao Han, Mingjie Zhan, Zhiyuan Huang, Delong Liu, Jiawei Guo, Zhicheng Zhao, and Fei Su. MOOM: maintenance, organization and optimization of memory in ultra-long role-playing dialogues. *CoRR*, abs/2509.11860, 2025.

[164] Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*, 2026.

[165] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *CoRR*, abs/2508.19828, 2025.

[166] Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025.

[167] Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

[169] Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, and Wangchunshu Zhou. O-mem: Omni memory system for personalized, long horizon, self-evolving agents, 2025.

[170] B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research, 2025.

[171] Mem0. Mem0: Universal memory layer for AI agents, 2026. GitHub repository.

[172] Mem0. OpenMemory: AI memory MCP server for coding agents, 2026. Product page.

[173] LangChain. Memory, 2026. Documentation.

[174] Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward, 2026.

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

[176] LangChain. Memory, 2026. Developer documentation.

[177] Anthropic. Equipping agents for the real world with agent skills, 2025. Blog post.

[178] OpenAI. Skills in openai api, 2026. Developer documentation.

[179] YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

[180] Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents, 2026.

[181] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[182] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, volume 2024, pages 54067–54089, 2024.

[183] Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, et al. Toolace: Winning the points of llm function calling. In *International Conference on Learning Representations*, volume 2025, pages 41359–41381, 2025.

[184] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024.

[185] Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[186] Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *CoRR*, abs/2504.07079, 2025.

[187] Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. Skillcraft: Can llm agents learn to use tools skillfully?, 2026.

[188] So Kuroki, Taishi Nakamura, Takuya Akiba, and Yujin Tang. Agent skill acquisition for large language models via cycleqd, 2025.

[189] Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration. In *The Fourteenth International Conference on Learning Representations*, 2026.

[190] Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Conference on Language Modeling (COLM)*, 2025.

[191] Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, Lianyong Qi, and Shi Jin. Contractskill: Repairable contract-based skills for multimodal web agents. *arXiv preprint arXiv:2603.20340*, 2026.

[192] Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. Hierarchical reinforcement learning: A survey and open research challenges. *Mach. Learn. Knowl. Extr.*, 4(1):172–221, 2022.

[193] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19632–19642. AAAI Press, 2024.

[194] Haochen Shi, Xingdi Yuan, and Bang Liu. Evolving programmatic skill networks. *CoRR*, abs/2601.03509, 2026.

[195] Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

[196] Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026.

[197] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

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

[199] Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

[200] Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026.

[201] Bingqing Wei, Zhongyu Xia, Dingai Liu, Xiaoyu Zhou, Zhiwei Lin, and Yongtao Wang. Elite: Experiential learning and intent-aware transfer for self-improving embodied agents. *arXiv preprint arXiv:2603.24018*, 2026.

[202] Max Wilcoxson, Qiyang Li, Kevin Frans, and Sergey Levine. Leveraging skills from unlabeled prior data for efficient online exploration. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 66833–66860. PMLR, 13–19 Jul 2025.

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

[204] Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. Skillfoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources. *arXiv preprint arXiv:2604.03964*, 2026.

[205] Anthropic. Agent skills, 2025. Developer documentation.

[206] Weihang Su, Jianming Long, Qingyao Ai, Yichen Tang, Changyue Wang, Yiteng Tu, and Yiqun Liu. Skill retrieval augmentation for agentic ai. *arXiv preprint arXiv:2604.24594*, 2026.

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

[208] Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. Skillflow: Scalable and efficient agent skill retrieval system. 2026.

[209] OpenAI. Agent skills in codex, 2026. Developer documentation.

[210] Dawei Li, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph of skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026.

[211] Fali Wang, Chenglin Weng, Xianren Zhang, Siyuan Hong, Hui Liu, and Suhang Wang. Graphskill: Documentation-guided hierarchical retrieval-augmented coding for complex graph reasoning. *arXiv preprint arXiv:2603.06620*, 2026.

[212] Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[213] Yijie Guo, Bingjie Tang, Iretiayo Akinola, Dieter Fox, Abhishek Gupta, and Yashraj Narang. SRSA: skill retrieval and adaptation for robotic assembly tasks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[214] Seoyoung Lee, Seobin Yoon, Seongbeen Lee, Yoojung Chun, Dayoung Park, Doyeon Kim, and Joo Yong Sim. Intentcua: Learning intent-level representations for skill abstraction and multi-agent planning in computer-use agents. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026)*, Paphos, Cyprus, May 2026.

[215] Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, et al. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026.

[216] Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. Evolver: Self-evolving LLM agents through an experience-driven lifecycle. *CoRR*, abs/2510.16079, 2025.

[217] OpenAI. Using skills to accelerate oss maintenance, 2026. Blog post.

[218] Charlie Guo. Shell + skills + compaction: Tips for long-running agents that do real work, 2026. Blog post.

[219] Dominik Kundel and Gabriel Chua. Testing agent skills systematically with evals, 2026. Blog post.

[220] Zhiyuan Li, Jingzheng Wu, Xiang Ling, Xing Cui, and Tianyue Luo. Towards secure agent skills: Architecture, threat taxonomy, and security analysis. *arXiv preprint arXiv:2604.02837*, 2026.

[221] Anthropic. How we built our multi-agent research system, 2025. Blog post.

[222] Qian Wang, Tianyu Wang, Zhenheng Tang, Qinbin Li, Nuo Chen, Jingsheng Liang, and Bingsheng He. MegaAgent: A large-scale autonomous LLM-based multi-agent system without predefined SOPs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4998–5036, Vienna, Austria, July 2025. Association for Computational Linguistics.

[223] Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *Advances in Neural Information Processing Systems*, 2025.

[224] Jiabin Tang, Tianyu Fan, and Chao Huang. Autoagent: A fully-automated and zero-code framework for llm agents. *arXiv preprint arXiv:2502.05957*, 2025.

[225] Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. In *Second Conference on Language Modeling*.

[226] Shanglin Wu, Yuyang Luo, Yueqing Liang, Kaiwen Shi, Yanfang Ye, Ali Payani, and Kai Shu. Scaling teams or scaling time? memory enabled lifelong learning in llm multi-agent systems, 2026.

[227] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In *NeurIPS*, 2023.

[228] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling (CoLM)*, 2024.

[229] Adnan Qayyum, Abdullatif Albaseer, Junaid Qadir, Ala Al-Fuqaha, and Mohamed Abdallah. Llm-driven multi-agent architectures for intelligent self-organizing networks. *IEEE Network*, pages 1–10, 2025.

[230] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. Proagent: Building proactive cooperative agents with large language models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17591–17599. AAAI Press, 2024.

[231] Priya Pitre, Naren Ramakrishnan, and Xuan Wang. CONSENSAGENT: Towards efficient and effective consensus in multi-agent LLM interactions through sycophancy mitigation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22112–22133, Vienna, Austria, July 2025. Association for Computational Linguistics.

[232] Anthropic. Building a c compiler with a team of parallel claudes, 2026. Blog post.

[233] Anthropic. Multi-agent coordination patterns: Five approaches and when to use them, 2026. Blog post.

[234] Cognition. Don't build multi-agents, 2025. Blog post.

[235] Noah Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli

Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *ACL (Findings)*, Findings of ACL, pages 14743–14777. Association for Computational Linguistics, 2024.

[236] Peiying Yu, Guoxin Chen, and Jingjing Wang. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. In *ACL*, pages 17432–17451. Association for Computational Linguistics, 2025.

[237] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *EMNLP*, pages 17889–17904. Association for Computational Linguistics, 2024.

[238] Zhe Hu, Hou Pong Chan, Jing Li, and Yu Yin. Debate-to-write: A persona-driven multi-agent framework for diverse argument generation. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4689–4703, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[239] Heewon Park and Minhae Kwon. Player-coach teamwork: Multi-agent collaboration for improving llm reasoning. In *NeurIPS 2025 Workshop on Scaling Environments for Agents (SEA)*, 2025.

[240] Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D Barrett, and Arnu Pretorius. Should we be going MAD? A look at multi-agent debate strategies for LLMs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 45883–45905. PMLR, 21–27 Jul 2024.

[241] Yurun Yuan and Tengyang Xie. Reinforce LLM reasoning through multi-agent reflection. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 73701–73731. PMLR, 13–19 Jul 2025.

[242] Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Brian Ichter, Anuj Mahajan, and Hamsa Balakrishnan. Long-horizon planning for multi-agent robots in partially observable environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 67929–67967. Curran Associates, Inc., 2024.

[243] Haotian Chi, Zeyu Feng, Yueming Lyu, Chengqi Zheng, Linbo Luo, Yew-Soon Ong, Ivor Tsang, Hechang Chen, Yi Chang, and Haiyan Yin. Instructflow: Adaptive symbolic constraint-guided code generation for long-horizon planning. In *Advances in Neural Information Processing Systems*, 2025.

[244] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. Flow-of-action: SOP enhanced llm-based multi-agent system for root cause analysis. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, April 28 - May 2, 2025*, pages 422–431. ACM, 2025.

[245] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62743–62767. PMLR, 21–27 Jul 2024.

[246] Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. Mapcoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4912–4944, 2024.

[247] Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *ACL (1)*, pages 30215–30248. Association for Computational Linguistics, 2025.

[248] Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: multi-agent reinforcement fine-tuning. *CoRR*, abs/2504.16129, 2025.

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

[250] Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, et al. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2508.13167*, 2025.

[251] Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating multi-agent intelligence with the tool-environment-agent(tea) protocol, 2026.

[252] Guoxin Chen, Jie Chen, Lei Chen, Jiale Zhao, Fanzhe Meng, Wayne Xin Zhao, Ruihua Song, Cheng Chen, Ji-Rong Wen, and Kai Jia. Toward autonomous long-horizon engineering for ml research. *arXiv preprint arXiv:2604.13018*, 2026.

[253] LangChain. How and when to build multi-agent systems, 2025. Blog post.

[254] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[255] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022.

[256] Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. *Nat.*, 623(7987):493–498, 2023.

[257] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *ACL/IJCNLP (1)*, pages 3816–3830. Association for Computational Linguistics, 2021.

[258] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

[259] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *ICML*, Proceedings of Machine Learning Research, pages 13481–13544. PMLR / OpenReview.net, 2024.

[260] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, 2024.

[261] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL (1)*, pages 10014–10037. Association for Computational Linguistics, 2023.

[262] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *EMNLP*, pages 7969–7992. Association for Computational Linguistics, 2023.

[263] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*. OpenReview.net, 2024.

[264] S Your et al. Crag: Corrective retrieval-augmented generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[265] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, 2024.

[266] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, 2025.

[267] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pages 32628–32649, 2024.

[268] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *NeurIPS*, 2024.

[269] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *ICLR*. OpenReview.net, 2024.

[270] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. *CoRR*, abs/2310.03714, 2023.

[271] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. A-RAG: scaling agentic retrieval-augmented generation via hierarchical retrieval interfaces. *CoRR*, abs/2602.03442, 2026.

[272] Thang Nguyen, Peter Chin, and Yu-Wing Tai. MA-RAG: multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *CoRR*, abs/2505.20096, 2025.

[273] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

[274] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *EMNLP*, pages 13358–13376. Association for Computational Linguistics, 2023.

[275] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *EMNLP*, pages 6342–6353. Association for Computational Linguistics, 2023.

[276] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. Llm maybe longlm: Selfextend llm context window without tuning. In *International Conference on Machine Learning*, pages 22099–22114. PMLR, 2024.

[277] Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: optimizing context compression for long-horizon LLM agents. *CoRR*, abs/2510.00615, 2025.

[278] Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management. *CoRR*, abs/2510.24699, 2025.

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

[280] Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents with interaction scaling. In *The Fourteenth International Conference on Learning Representations*, 2026.

[281] Rishiraj Saha Roy, Chris Hinze, Luzian Hahn, and Fabian Kuech. Cedar: Context engineering for agentic data science. *arXiv preprint arXiv:2601.06606*, 2026.

[282] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*. OpenReview.net, 2024.

[283] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[284] Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. *arXiv preprint arXiv:2601.21557*, 2026.

[285] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *ACL (1)*, pages 32779–32798. Association for Computational Linguistics, 2025.

[286] Junfeng Lu and Yueyan Li. Dynamic affective memory management for personalized LLM agents. *CoRR*, abs/2510.27418, 2025.

[287] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*. OpenReview.net, 2021.

[288] Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *EMNLP*, pages 11279–11298. Association for Computational Linguistics, 2022.

[289] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

[290] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[291] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7889–7903, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[292] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning LLM from zero data. *CoRR*, abs/2508.05004, 2025.

[293] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example. *CoRR*, abs/2504.20571, 2025.

[294] Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. Formalmath: Benchmarking formal mathematical reasoning of large language models. *CoRR*, abs/2505.02735, 2025.

[295] Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob N. Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. Mlgym: A new framework and benchmark for advancing AI research agents. *CoRR*, abs/2502.14499, 2025.

[296] John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In *NeurIPS*, 2023.

[297] Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. REASONING GYM: reasoning environments for reinforcement learning with verifiable rewards. *CoRR*, abs/2505.24760, 2025.

[298] Shiva Krishna Reddy Malay, Shravan Nayak, Jishnu Sethumadhavan Nair, Sagar Davasam, Aman Tiwari, Sathwik Tejaswi Madhusudhan, Sridhar Krishna Nemala, Srinivas Sunkara, and Sai Rajeswar. Enterpriseops-gym: Environments and evaluations for stateful agentic planning and tool use in enterprise settings. *CoRR*, abs/2603.13594, 2026.

[299] Jiale Zhao, Guoxin Chen, Fanzhe Meng, Minghao Li, Jie Chen, Hui Xu, Yongshuai Sun, Xin Zhao, Ruihua Song, Yuan Zhang, Peng Wang, Cheng Chen, Jirong Wen, and Kai Jia. Immersion in the github universe: Scaling coding agents to mastery. *CoRR*, abs/2602.09892, 2026.

[300] Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. Webworld: A large-scale world model for web agent training. *CoRR*, abs/2602.14721, 2026.

[301] Luke Rivard, Sun Sun, Hongyu Guo, Wenhu Chen, and Yuntian Deng. Neuralos: Towards simulating operating systems via neural generative models. *CoRR*, abs/2507.08800, 2025.

[302] Raj Ghugare, Catherine Ji, Kathryn Wantlin, Jin Schofield, and Benjamin Eysenbach. Builderbench - A benchmark for generalist agents. *CoRR*, abs/2510.06288, 2025.

[303] Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Bram Grooten, Meng Fang, and Mykola Pechenizkiy. MEAL: A benchmark for continual multi-agent reinforcement learning. *CoRR*, abs/2506.14990, 2025.

[304] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *EMNLP*, pages 8154–8173. Association for Computational Linguistics, 2023.

[305] Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *CoRR*, abs/2602.10090, 2026.

[306] Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Trans. Mach. Learn. Res.*, 2025, 2025.

[307] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *ICLR*. OpenReview.net, 2024.

[308] Hao Bai, Yifei Zhou, Li Erran Li, Sergey Levine, and Aviral Kumar. Digi-q: Learning VLM q-value functions for training device-control agents. In *ICLR*. OpenReview.net, 2025.

[309] Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *ICLR*. OpenReview.net, 2025.

[310] Tanmay Gupta, Piper Wolters, Zixian Ma, Peter Sushko, Rock Yuren Pang, Diego Llanes, Yue Yang, Taira Anderson, Boyuan Zheng, Zhongzheng Ren, Harsh Trivedi, Taylor Blanton, Caleb Ouellette, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmoweb: Open visual web agent and open data for the open web. *CoRR*, abs/2604.08516, 2026.

[311] Sushant Mehta, Logan Ritchie, Suhaas Garre, Ian Niebres, Nick Heiner, and Edwin Chen. Enterprisebench corecraft: Training generalizable agents on high-fidelity RL environments. *CoRR*, abs/2602.16179, 2026.

[312] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *EMNLP*, pages 414–431. Association for Computational Linguistics, 2025.

[313] Huanyao Zhang, Jiepeng Zhou, Bo Li, Bowen Zhou, Yanzhe Shan, Haishan Lu, Zhiyong Cao, Jiaoyang Chen, Yuqian Han, Zinan Sheng, Zhengwei Tao, Hao Liang, Jialong Wu, Yang Shi, Yuanpeng He, Jiaye Lin, Qintong Zhang, Guochen Yan, Runhao Zhao, Zhengpin Li, Xiaohan Yu, Lang Mei, Chong Chen, Wentao Zhang, and Bin Cui. Browsecomp-v^3: A visual, vertical, and verifiable benchmark for multimodal browsing agents. *CoRR*, abs/2602.12876, 2026.

[314] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *ICML*, Proceedings of Machine Learning Research, pages 8469–8488. PMLR, 2023.

[315] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In *CoRL*, Proceedings of Machine Learning Research, pages 2165–2183. PMLR, 2023.

[316] Zhihao Wang, Jianxiong Li, Jinliang Zheng, Wencong Zhang, Dongxiu Liu, Yinan Zheng, Haoyi Niu, Junzhi Yu, and Xianyuan Zhan. Physiagent: An embodied agent framework in physical world. *CoRR*, abs/2509.24524, 2025.

[317] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023.

[318] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *ACL (Findings)*, Findings of ACL, pages 3053–3077. Association for Computational Linguistics, 2024.

[319] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *ACL (Findings)*, Findings of ACL, pages 9354–9366. Association for Computational Linguistics, 2024.

[320] Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, Tulika Manoj Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, and Caiming Xiong. Agentohana: Design unified data and training pipeline for effective agent learning. *CoRR*, abs/2402.15506, 2024.

[321] Xuanzhong Chen, Zile Qiao, Guoxin Chen, Liangcai Su, Zhen Zhang, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Yong Jiang, and Ting Chen. Expanding the capability frontier of LLM agents with ZPD-guided data synthesis. In *The Fourteenth International Conference on Learning Representations*, 2026.

[322] Renjie Pi, Grace Lam, Mohammad Shoeybi, Pooya Jannaty, Bryan Catanzaro, and Wei Ping. On data engineering for scaling LLM terminal capabilities. *CoRR*, abs/2602.21193, 2026.

[323] Yanheng He, Jiahe Jin, and Pengfei Liu. Efficient agent training for computer use. *CoRR*, abs/2505.13909, 2025.

[324] Fei Bai, Huatong Song, Shuang Sun, Daixuan Cheng, Yike Yang, Chuan Hao, Renyuan Li, Feng Chang, Yuan Wei, Ran Tao, Bryan Dai, Jian Yang, and Wayne Xin Zhao. Clawgym: A scalable framework for building effective claw agents, 2026.

[325] Zhanming Shen, Zeyu Qin, Zenan Huang, Hao Chen, Jiaqi Hu, Yihong Zhuang, Guoshan Lu, Gang Chen, and Junbo Zhao. Merge-of-thought distillation. *CoRR*, abs/2509.08814, 2025.

[326] Justin Chih-Yao Chen, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. Magdi: Structured distillation of multi-agent interaction graphs improves reasoning in smaller language models. In *ICML*, Proceedings of Machine Learning Research, pages 7220–7235. PMLR / OpenReview.net, 2024.

[327] Jun Liu, Zhenglun Kong, Peiyan Dong, Changdi Yang, Tianqi Li, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Pu Zhao, Xue Lin, Dong Huang, and Yanzhi Wang. Structured agent distillation for large language model. *CoRR*, abs/2505.13820, 2025.

[328] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*. OpenReview.net, 2024.

[329] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*. OpenReview.net, 2024.

[330] Ijun Jang, Je Won Yeom, Juan Yeo, Hyunggu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. *CoRR*, abs/2601.07155, 2026.

[331] Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. *arXiv preprint arXiv:2603.11137*, 2026.

[332] Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *CoRR*, abs/2602.12125, 2026.

[333] Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *CoRR*, abs/2511.10643, 2025.

[334] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[335] Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of RLHF in large language models part I: PPO. *CoRR*, abs/2307.04964, 2023.

[336] Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Secrets of RLHF in large language models part II: reward modeling. *CoRR*, abs/2401.06080, 2024.

[337] Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. WARP: on the benefits of weight averaged rewarded policies. *CoRR*, abs/2406.16768, 2024.

[338] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. In *ICML*, Proceedings of Machine Learning Research, pages 62178–62209. PMLR / OpenReview.net, 2024.

[339] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62138–62160, 2024.

[340] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron C. Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in RL training of llms. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[341] Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. In *AAAI*, pages 32150–32158. AAAI Press, 2026.

[342] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

[343] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: model alignment as prospect theoretic optimization. *CoRR*, abs/2402.01306, 2024.

[344] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In *NeurIPS*, 2024.

[345] Jiwoo Hong, Noah Lee, and James Thorne. ORPO: monolithic preference optimization without reference model. In *EMNLP*, pages 11170–11189. Association for Computational Linguistics, 2024.

[346] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

[347] Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *ICML*, Proceedings of Machine Learning Research, pages 29128–29163. PMLR / OpenReview.net, 2024.

[348] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978, 2025.

[349] Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. *CoRR*, abs/2509.21240, 2025.

[350] Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025.

[351] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849, 2025.

[352] Zexi Liu, Jingyi Chai, Xinyu Zhu, Shuo Tang, Rui Ye, Bolun Zhang, Lei Bai, and Siheng Chen. Ml-agent: Reinforcing LLM agents for autonomous machine learning engineering. *CoRR*, abs/2505.23723, 2025.

[353] Weize Liu, Minghui Liu, Sy-Tuyen Ho, Souradip Chakraborty, Xiyao Wang, and Furong Huang. Agentic critical training. *arXiv preprint arXiv:2603.08706*, 2026.

[354] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *CoRR*, abs/2505.01441, 2025.

[355] Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agent RL scaling law: Agent RL with spontaneous code execution for mathematical problem solving. *CoRR*, abs/2505.07773, 2025.

[356] Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: acing coder RL via automated test-case synthesis. In *ACL (1)*, pages 12023–12040. Association for Computational Linguistics, 2025.

[357] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level. *Notion Blog*, 1, 2025.

[358] Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, et al. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl, 2025. *Notion Blog*, 1, 2025.

[359] Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[360] Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

[361] Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. TTRL: test-time reinforcement learning. *CoRR*, abs/2504.16084, 2025.

[362] Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *ICLR*. OpenReview.net, 2025.

[363] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*. OpenReview.net, 2024.

[364] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL (1)*, pages 9426–9439. Association for Computational Linguistics, 2024.

[365] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592, 2024.

[366] Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g^2: Gaussian reward modeling for GUI grounding. In *AAAI*, pages 33214–33222. AAAI Press, 2026.

[367] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Trans. Mach. Learn. Res.*, 2026, 2026.

[368] Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve LLM agents for sequential decision-making tasks. *CoRR*, abs/2505.00234, 2025.

[369] Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance RLHF framework. *CoRR*, abs/2405.11143, 2024.

[370] Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huanchen Zhang, and Yi Wu. Realhf: Optimized RLHF training for large language models through parameter reallocation. *CoRR*, abs/2406.14088, 2024.

[371] Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy J. Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal, and Oleksii Kuchaiev. Nemo-aligner: Scalable toolkit for efficient model alignment. *CoRR*, abs/2405.01481, 2024.

[372] Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *CoRR*, abs/2505.24298, 2025.

[373] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *CoRR*, abs/2503.24290, 2025.

[374] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments. *CoRR*, abs/2406.04151, 2024.

[375] Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, Chuen Yang Beh, Weixun Wang, Hao Zhu, Weiyan Shi, Diyi Yang, Michael Shieh, Yee Whye Teh, Wee Sun Lee, and Min Lin. GEM: A gym for agentic llms. *CoRR*, abs/2510.01051, 2025.

[376] Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train ANY AI agents with reinforcement learning. *CoRR*, abs/2508.03680, 2025.

[377] Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. *CoRR*, abs/2509.08755, 2025.

[378] Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *CoRR*, abs/2510.04206, 2025.

[379] Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. Agent-r1: Training powerful LLM agents with end-to-end reinforcement learning. *CoRR*, abs/2511.14460, 2025.

[380] Wei Gao, Yuheng Zhao, Tianyuan Wu, Shaopan Xiong, Weixun Wang, Dakai An, Lunxi Cao, Dilxat Muhtar, Zichen Liu, Haizhou Zhao, Ju Huang, Siran Yang, Yongbin Li, Wenbo Su,

Jiamang Wang, Lin Qu, Bo Zheng, and Wei Wang. Rollart: Scaling agentic RL training via disaggregated infrastructure. *CoRR*, abs/2512.22560, 2025.

[381] Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-agent: Efficient RL training for multi-turn LLM agent. *CoRR*, abs/2511.16108, 2025.

[382] Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: stable reinforcement learning for multi-agent LLM systems. *CoRR*, abs/2602.08847, 2026.

[383] Kaiyan Zhang, Kai Tian, Runze Liu, Sihang Zeng, Xuekai Zhu, Guoli Jia, Yuchen Fan, Xingtai Lv, Yuxin Zuo, Che Jiang, et al. Marti: A framework for multi-agent llm systems reinforced training and inference. In *The Fourteenth International Conference on Learning Representations*, 2026.

[384] Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. Wideseek-r1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *CoRR*, abs/2602.04634, 2026.

[385] DeepReinforce Team, Xiaoya Li, Xiaofei Sun, Guoyin Wang, Songqiao Su, Chris Shum, and Jiwei Li. Grandcode: Achieving grandmaster level in competitive programming via agentic reinforcement learning. *CoRR*, abs/2604.02721, 2026.

[386] Reuben Tan, Baolin Peng, Zhengyuan Yang, Hao Cheng, Oier Mees, Theodore Zhao, Andrea Tupini, Isar Meijier, Qianhui Wu, Yuncong Yang, Lars Liden, Yu Gu, Sheng Zhang, Xiaodong Liu, Lijuan Wang, Marc Pollefeys, Yong Jae Lee, and Jianfeng Gao. Multimodal reinforcement learning with agentic verifier for AI agents. *CoRR*, abs/2512.03438, 2025.

[387] Wanli Li, Bince Qu, Bo Pan, Jianyu Zhang, Zheng Liu, Pan Zhang, Wei Chen, and Bo Zhang. Literesearcher: A scalable agentic rl training framework for deep research agent. *arXiv preprint arXiv:2604.17931*, 2026.

[388] Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. Mm-deepresearch: A simple and effective multimodal agentic search baseline. *CoRR*, abs/2603.01050, 2026.

[389] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vázquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 11642–11662. PMLR / OpenReview.net, 2024.

[390] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024.

[391] Yingchaojie Feng, Qiang Huang, Xiaoya Xie, Zhaorui Yang, Jun Yu, Wei Chen, and Anthony K. H. Tung. Idrbench: Interactive deep research benchmark. *CoRR*, abs/2601.06676, 2026.

[392] Ruizhe Li, Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench II: diagnosing deep research agents via rubrics from expert report. *CoRR*, abs/2601.08536, 2026.

[393] Benjamin Elder, Anupama Murthi, Jungkoo Kang, Ankita Naik, Kinjal Basu, Kiran Kate, and Danish Contractor. Live api-bench: 2500+ live apis for testing multi-step tool calling. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2026 - Volume 1: Long Papers, Rabat, Morocco, March 24-29, 2026*, pages 3092–3124. Association for Computational Linguistics, 2026.

[394] Yifu Cai, Xinyu Li, Mononito Goswami, Michal Wilinski, Gus Welter, and Artur Dubrawski. Timeseriesgym: A scalable benchmark for (time series) machine learning engineering agents. *CoRR*, abs/2505.13291, 2025.

[395] Minghao Li, Ying Zeng, Zhihao Cheng, Cong Ma, and Kai Jia. Reportbench: Evaluating deep research agents via academic survey tasks. *CoRR*, abs/2508.15804, 2025.

[396] Jiayu Wang, Yifei Ming, Riya Dulepet, Qinglin Chen, Austin Xu, Zixuan Ke, Frederic Sala, Aws Albarghouthi, Caiming Xiong, and Shafiq Joty. Liveresearchbench: A live benchmark for user-centric deep research in the wild. *CoRR*, abs/2510.14240, 2025.

[397] Yuan Liang, Jiaxian Li, Yuqing Wang, Piaohong Wang, Motong Tian, Pai Liu, Shuofei Qiao, Runnan Fang, He Zhu, Ge Zhang, Minghao Liu, Yuchen Eleanor Jiang, Ningyu Zhang, and Wangchunshu Zhou. Towards personalized deep research: Benchmarks and evaluations. *CoRR*, abs/2509.25106, 2025.

[398] Chengwen Liu, Xiaomin Yu, Zhuoyue Chang, Zhe Huang, Shuo Zhang, Heng Lian, Kunyi Wang, Rui Xu, Sen Hu, Jianheng Hou, Hao Peng, Chengwei Qin, Xiaobin Hu, Hong Peng, Ronghao Chen, and Huacan Wang. Watching, reasoning, and searching: A video deep research benchmark on open web for agentic video reasoning. *CoRR*, abs/2601.06943, 2026.

[399] Peizhou Huang, Zixuan Zhong, Zhongwei Wan, Donghao Zhou, Samiul Alam, Xin Wang, Zexin Li, Zhihao Dou, Li Zhu, Jing Xiong, Chaofan Tao, Yan Xu, Dimitrios Dimitriadis, Tuo Zhang, and Mi Zhang. Mmdeepresearch-bench: A benchmark for multimodal deep research agents. *CoRR*, abs/2601.12346, 2026.

[400] Manasi Sharma, Chen Bo Calvin Zhang, Chaithanya Bandi, Clinton Wang, Ankit Aich, Huy Nghiem, Tahseen Rabbani, Ye Htet, Brian Jang, Sumana Basu, Aishwarya Balwani, Denis Peskoff, Marcos Ayestaran, Sean M. Hendryx, Brad Kenstler, and Bing Liu. Researchrubrics: A benchmark of prompts and rubrics for evaluating deep research agents. *CoRR*, abs/2511.07685, 2025.

[401] Tianyang Liu, Canwen Xu, and Julian J. McAuley. Repobench: Benchmarking repository-level code auto-completion systems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[402] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13643–13658. Association for Computational Linguistics, 2024.

[403] Jielin Qiu, Zuxin Liu, Zhiwei Liu, Rithesh Murthy, Jianguo Zhang, Haolin Chen, Shiyu Wang, Ming Zhu, Liangwei Yang, Juntao Tan, Roshan Ram, Akshara Prabhakar, Tulika Awalgaonkar, Zixiang Chen, Zhepeng Cen, Cheng Qian, Shelby Heinecke, Weiran Yao, Silvio Savarese, Caiming Xiong, and Huan Wang. Locobench-agent: An interactive benchmark for LLM agents in long-context software engineering. *CoRR*, abs/2511.13998, 2025.

[404] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[405] Sanket Mhatre, Yasharth Bajpai, Sumit Gulwani, Emerson R. Murphy-Hill, and Gustavo Soares. Swe-sharp-bench: A reproducible benchmark for c# software engineering tasks. In *2nd IEEE/ACM International Conference on AI-powered Software, AIware 2025, Seoul, Republic of Korea, November 19-20, 2025*, pages 277–280. IEEE, 2025.

[406] Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, Weiran Shi, Zaiyuan Wang, Daoguang Zan, Chenchen Zhang, Xiaoxu Zhang, Qizhi Chen, Xianfu Cheng, Bo Deng, Qingshui Gu, Kai Hua, Juntao Lin, Pai Liu, Mingchen Li, Xuanguang Pan, Zifan Peng, Yujia Qin, Yong Shan, Zhewen Tan, Weihao Xie, Zihan Wang, Yishuo Yuan, Jiayu Zhang, Enduo Zhao, Yunfei Zhao, He Zhu, Chenyang Zou, Ming Ding, Jianpeng Jiao, Jiaheng Liu, Minghao Liu, Qian Liu, Chongyao Tao, Jian Yang, Tong Yang, Zhaoxiang Zhang, Xinjie Chen, Wenhao Huang, and Ge Zhang. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents. *CoRR*, abs/2512.12730, 2025.

[407] Qixing Zhou, Jiacheng Zhang, Haiyang Wang, Rui Hao, Jiahe Wang, Minghao Han, Yuxue Yang, Shuzhe Wu, Feiyang Pan, Lue Fan, Dandan Tu, and Zhaoxiang Zhang. Featurebench: Benchmarking agentic coding for complex feature development. *CoRR*, abs/2602.10975, 2026.

[408] Guoxin Chen, Fanzhe Meng, Jiale Zhao, Minghao Li, Daixuan Cheng, Huatong Song, Jie Chen, Yuzhi Lin, Hui Chen, Xin Zhao, et al. Beyondswe: Can current code agent survive beyond single-repo bug fixing? *arXiv preprint arXiv:2603.03194*, 2026.

[409] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics, 2023.

[410] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[411] Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. Assistantbench: Can web agents solve realistic and time-consuming tasks? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8938–8968. Association for Computational Linguistics, 2024.

[412] Junjie Ye, Guoqiang Zhang, Wenjie Fu, Tao Gui, Qi Zhang, and Xuanjing Huang. CCTU: A benchmark for tool use under complex constraints. *CoRR*, abs/2603.15309, 2026.

[413] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P. Lillicrap. Android in the wild: A large-scale dataset for android device control. *CoRR*, abs/2307.10088, 2023.

[414] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows agent arena: Evaluating multi-modal OS agents at scale. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[415] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: GUI grounding for professional high-resolution computer use. In Cathal Gurrin, Klaus Schoeffmann, Min Zhang, Luca Rossetto, Stevan Rudinac, Duc-Tien Dang-Nguyen, Wen-Huang Cheng, Phoebe Chen, and Jenny Benois-Pineau, editors, *Proceedings of the 33rd ACM International Conference on Multimedia, MM 2025, Dublin, Ireland, October 27-31, 2025*, pages 8778–8786. ACM, 2025.

[416] Beitong Zhou, Zhexiao Huang, Yuan Guo, Zhangxuan Gu, Tianyu Xia, Zichen Luo, Fei Tang, Dehan Kong, Yanyi Shang, Suling Ou, Zhenlin Guo, Changhua Meng, and Shuheng Shen. Venusbench-gd: A comprehensive multi-platform GUI benchmark for diverse grounding tasks. *CoRR*, abs/2512.16501, 2025.

[417] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida I. Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 18319–18345. PMLR, 2023.

[418] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 20271–20309. PMLR / OpenReview.net, 2024.

[419] Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and Kang Liu. Da-code: Agent data science code generation benchmark for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language*

Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024, pages 13487–13521. Association for Computational Linguistics, 2024.

[420] Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating AI agents on open-ended machine learning research. *CoRR*, abs/2505.19955, 2025.

[421] Qiran Zou, Hein-Mun Lam, Wenhao Zhao, Yiming Tang, Tingting Chen, Samson Yu, Tianyi Zhang, Chang Liu, Xiangyang Ji, and Dianbo Liu. Fml-bench: A benchmark for automatic ML research agents highlighting the importance of exploration breadth. *CoRR*, abs/2510.10472, 2025.

[422] Bang Nguyen, Dominik Soós, Qian Ma, Rochana R. Obadage, Zack Ranjan, Sai Dileep Koneru, Timothy M. Errington, Shakhlo Nematova, Sarah Rajtmajer, Jian Wu, and Meng Jiang. Replicatorbench: Benchmarking LLM agents for replicability in social and behavioral sciences. *CoRR*, abs/2602.11354, 2026.

[423] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025.

[424] Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. Chain of thoughtlessness? an analysis of cot in planning. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[425] Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Reducing cost of llm agents with trajectory reduction. *Proceedings of the ACM on Software Engineering*, 3(FSE), 2026. arXiv preprint arXiv:2509.23586.

[426] Yilun Yao, Shan Huang, Elsie Dai, Zhewen Tan, Zhenyu Duan, Shousheng Jia, Yanbing Jiang, and Tong Yang. ARC: active and reflection-driven context management for long-horizon information seeking agents. *CoRR*, abs/2601.12030, 2026.

[427] Shukai Liu, Jian Yang, Bo Jiang, Yizhi Li, Jinyang Guo, Xianglong Liu, and Bryan Dai. Context as a tool: Context management for long-horizon swe-agents. *CoRR*, abs/2512.22087, 2025.

[428] Tao Zhe, Haoyu Wang, Bo Luo, Min Wu, Wei Fan, Xiao Luo, Zijun Yao, Haifeng Chen, and Dongjie Wang. Robust and efficient tool orchestration via layered execution structures with reflective correction. *arXiv preprint arXiv:2602.18968*, 2026.

[429] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 82895–82920. Curran Associates, Inc., 2024.

[430] Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. Mcptox: A benchmark for tool poisoning attack on real-world mcp servers, 2025.

[431] Genliang Zhu, Chu Wang, Ziyuan Wang, Zhida Li, and Qiang Li. Openport protocol: A security governance specification for ai agent tool access. *arXiv preprint arXiv:2602.20196*, 2026.

[432] Maurits Kaptein, Vassilis-Javed Khan, and Andriy Podstavnychy. Runtime governance for ai agents: Policies on paths. *arXiv preprint arXiv:2603.16586*, 2026.

[433] Haoyu Wang, Christopher M Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable llm agents.(2026). In *Proceedings of the IEEE/ACM International Conference on Software Engineering, ICSE*, pages 12–18, 2026.

[434] Herman Errico. Autonomous action runtime management(aarm):a system specification for securing ai-driven actions at runtime. *CoRR*, abs/2602.09433, 2026.

[435] Xixun Lin, Yang Liu, Yancheng Chen, Yongxuan Wu, Yucheng Ning, Yilong Liu, Nan Sun, Shun Zhang, Bin Chong, Chuan Zhou, and Yanan Cao. Safeharness: Lifecycle-integrated security architecture for llm-based agent deployment, 2026.

[436] Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents, 2026.

[437] Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. *CoRR*, abs/2510.12635, 2025.

[438] Isabella He. Context engineering: Memory, compaction, and tool clearing, 2026. Blog post.

[439] Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[440] Haitao Jiang, Lin Ge, Hengrui Cai, and Rui Song. Pabu: Progress-aware belief update for efficient llm agents, 2026.

[441] Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence, 2026.

[442] I. Apanasevich, M. Artemyev, R. Babakyan, P. Fedotova, Denis Grankin, E. Kupryashin, A. Misailidi, D. Nerus, A. Nutalapati, G. Sidorov, I. Efremov, M. Gerasyov, D. Pikurov, Y. Senchenko, Sergei Davidenko, D. Kulikov, M. Sultankin, K. Askarbek, O. Shamanin, Dmitrii Statovoy, E. Zalyaev, I. Zorin, A. Letkin, E. Rusakov, A. Silchenko, V. Vorobyov, S. Sobolnikov, and Aleksey Postnikov. Green-vla: Staged vision-language-action model for generalist robots. *CoRR*, abs/2602.00919, 2026.

[443] Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. *CoRR*, abs/2601.11404, 2026.

[444] Jinhui Ye, Fangjing Wang, Ning Gao, Junqiu Yu, Yangkun Zhu, Bin Wang, Jinyu Zhang, Weiyang Jin, Yanwei Fu, Feng Zheng, Yilun Chen, and Jiangmiao Pang. ST4VLA: spatially guided training for vision-language-action models. *CoRR*, abs/2602.10109, 2026.

[445] Peiran Xu, Jiaqi Zheng, and Yadong Mu. Roboagent: Chaining basic capabilities for embodied task planning, 2026.

[446] Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable LLM agents. *CoRR*, abs/2503.18666, 2025.

[447] Chenxin Li, Zhengyang Tang, Mingxin Huang, Yunlong Lin, Shijue Huang, Shengyuan Liu, Bowen Ye, Rang Li, Lei Li, Benyou Wang, and Yixuan Yuan. Claw-eval-live: A live agent benchmark for evolving real-world workflows, 2026.

[448] Shicheng Fang, Yuxin Wang, Xiaoran Liu, Jiahao Lu, Chuanyuan Tan, Xinchi Chen, Yining Zheng, Xuanjing Huang, and Xipeng Qiu. Agentlongbench: A controllable long benchmark for long-contexts agents via environment rollouts, 2026.

[449] Yu Li, Haoyu Luo, Yuejin Xie, Yuqian Fu, Zhonghao Yang, Shuai Shao, Qihan Ren, Wanying Qu, Yanwei Fu, Yujiu Yang, Jing Shao, Xia Hu, and Dongrui Liu. Atbench: A diverse and realistic agent trajectory benchmark for safety evaluation and diagnosis, 2026.

[450] Faouzi El Yagoubi, Godwin Badu-Marfo, and Ranwa Al Mallah. Agentleak: A full-stack benchmark for privacy leakage in multi-agent llm systems, 2026.

[451] Xiaochuan Li, Ryan Ming, Pranav Setlur, Abhijay Paladugu, Andy Tang, Hao Kang, Shuai Shao, Rong Jin, and Chenyan Xiong. Benchmark test-time scaling of general llm agents, 2026.