agent latex survey

February 7, 2026

Contents

Abstract

Tool-using large language model (LLM) agents are increasingly deployed as closed-loop systems that plan, act through external tools, and adapt to feedback in dynamic environments. Their reported gains are often difficult to interpret in isolation because results conflate the agent loop, the tool/interface contract, budget assumptions (steps, cost, latency), and threat models for tool misuse. This survey organizes the design space around (i) foundations and interfaces, (ii) core components for long-horizon behavior (planning and memory), and (iii) learning, coordination, evaluation, and risks. We synthesize representative patterns such as reasoning-action prompting, tool-call induction, reflection, and deliberate search, and connect them to benchmark suites and protocol choices that determine which comparisons are meaningful. Across these lenses, we highlight recurring failure modes - prompt injection, tool abuse, and evaluation leakage - and outline a protocol-aware checklist for reporting and auditing agent results. [137, 99, 106, 138, 83, 28]

1 Introduction

Tool-using LLM agents turn language models from passive generators into controllers of action: they decide what to do next, invoke external tools, read observations, and iterate until a task terminates. This closed-loop view explains both the recent leap in practical capability and the persistent confusion in interpreting results: improvements can come from better prompting, better tools, better interface contracts, or simply different budgets and stopping rules. Canonical agent patterns such as reasoning-action prompting, tool-call induction, reflection, and deliberate search already illustrate this entanglement, even when evaluated on the same interactive tasks. [137, 99, 106, 138]

An operational definition used throughout this paper is: an agent is a system that (i) maintains state, (ii) selects actions in an action space that includes tool/API calls or environment operations, and (iii) updates state from observations to continue acting under an explicit protocol. This definition separates agent behavior from one-shot tool augmentation (e.g., a single retrieval call) and focuses attention on what can fail in practice: action representation, observation fidelity, and recovery strategies when tools are unreliable. Work on embodied or long-horizon settings further highlights that agents accumulate skills and abstractions over time, making the interface between the model, memory, and tools a first-class design choice rather than an implementation detail. [119, 22, 109]

The first organizing axis is the interface contract: what tools exist, how they are described, what arguments are legal, what the tool returns, and what permissions and logging surround execution. Two systems can share the same base model but behave very differently if one has a stable schema with sandboxed execution while the other relies on underspecified natural-language tool descriptions. Recent benchmark and security analyses show that interface design is inseparable from reliability and risk, including prompt injection pathways and tool-side manipulation that do not appear in text-only evaluations. [70, 32, 84, 150]

The second axis is the agent loop itself: how decisions are formed and revised as new evidence arrives. Planning and reasoning loops control when an agent commits to an action versus exploring alternatives, and they interact strongly with budget constraints (latency, cost, step limits) that are often underreported. A practical implication is that protocol-aware comparisons should treat planning as an algorithmic component with an evaluation footprint, not merely as a prompt style, because changes in search breadth or tool-call frequency can dominate measured outcomes. [138, 101, 30, 86]

Memory and retrieval provide the third axis: what information an agent can condition on, how it is selected, and how it is verified. Retrieval-augmented agents can appear strong on benchmarks where fresh evidence is available, yet fragile when retrieval introduces spurious citations, stale context, or adversarial content. A recurring theme across agent systems is that memory modules are not interchangeable; their failure modes depend on indexing choices, update policies, and how retrieval results are integrated into the loop (as constraints, as suggestions, or as authoritative facts). [103, 1, 64, 18]

Learning and adaptation mechanisms add another layer of coupling between interface and behavior. Reflection and self-improvement loops can change not only the final answer but also tool usage patterns (which tools are called, how often, and with what arguments), complicating attribution when improvements are measured as task success alone. Multi-agent variants further amplify these effects: role specialization and communication protocols introduce new degrees of freedom, while aggregation rules can hide individual agent failures behind ensemble outcomes. [106, 167, 25, 72, 73]

Evaluation is therefore not just a downstream reporting step; it defines what claims are meaningful. Benchmark suites for tool-use agents and agentic systems differ in task families, scoring rules, interaction formats, and threat models, and comparisons are often brittle unless protocol fields (tool access, budgets, termination, and safety constraints) are normalized. Recent evaluations emphasize this protocol gap by cataloging benchmark landscapes and by introducing suites that stress realistic tool definitions, industrial workflows, and adversarial settings. [83, 15, 87, 144, 28]

Security and governance constraints are increasingly central because the action space itself creates new attack surfaces: prompt injection, data exfiltration through tools, and deceptive tool outputs can break the implicit assumptions behind standard success metrics. Defenses often trade off robustness and utility, and the threat model (what the adversary controls, what the agent can call, what is logged) changes the interpretation of any reported improvement. A key motivation for a protocol-aware survey is to connect these risk considerations to the same interface and loop choices that drive performance. [32, 162, 3, 53, 82]

Methodology. We retrieved a candidate pool of 1,800 papers and curated a 300-paper core set for synthesis; unless otherwise noted, evidence is drawn from abstracts and metadata rather than full-text extraction. This evidence level is sufficient to map the design space and protocols, but it can underrepresent implementation-specific details; accordingly, the survey emphasizes protocol fields (interfaces, budgets, threat models) that are typically stated explicitly, and it flags where deeper verification is most likely to change interpretation. [91, 63, 35, 133]

The remainder of the paper follows the above lenses. Section 3 frames foundations and interfaces by separating the agent loop from the action space and the tool contract. Section 4 focuses on core components for long-horizon behavior, with planning and memory treated as protocol-sensitive modules. Section 5 covers learning, adaptation, and multi-agent coordination as mechanisms that alter both behavior and tool usage. Section 6 surveys evaluation and risks, emphasizing benchmark design and security threat models. Appendix tables summarize representative loop patterns and evaluation settings to make protocol assumptions explicit. [137, 99, 83, 150]

2 Related Work

Related work for LLM agents spans three overlapping categories: (i) surveys and taxonomies that map the broader agentic landscape, (ii) system and method papers that define agent loops, tool interfaces, planning, and memory, and (iii) evaluation and security work that establishes benchmarks, protocols, and threat models. A recurring limitation across these streams is that comparisons are often presented without fully normalizing protocol fields such as tool access, budgets, and safety constraints, which makes head-to-head interpretation fragile even when papers appear to address similar tasks. [83, 15, 28]

Surveys of agentic LLMs provide useful overviews of common architectures and application domains, including tool use, planning, feedback learning, and coordination. They typically emphasize the breadth of agent behaviors and the rapid growth of systems, but they vary in how explicitly they represent protocol assumptions and how tightly they connect evaluation results to interface contracts. The present work builds on these mappings and focuses on making protocol fields and interface assumptions explicit as the primary lens for synthesis. [91, 63, 35, 133, 83]

Foundational method papers define several reusable agent loop templates that continue to anchor system design. Reasoning-action prompting formalizes iterative decision making with tool calls, while tool-call induction and reflection-based loops treat tool usage and revision as learnable behaviors rather than fixed prompting conventions. Deliberate search approaches frame planning as exploration in a space of intermediate reasoning states, which interacts strongly with budgets and stopping criteria in interactive tasks. These lines of work motivate treating the loop design and interface contract as coupled objects when comparing agent performance. [137, 99, 106, 138]

Tool interface and orchestration papers highlight that the action space is not merely a set of tools, but also a contract about schemas, permissions, and observability. Benchmarks built around realistic tool definitions and execution environments stress that tool-use evaluation must model failure recovery, argument correctness, and the interaction format between the agent and tools. Dataset-driven approaches further suggest that tool usage patterns can be improved via scale, but only when tools are standardized and the execution harness is stable enough to provide meaningful feedback signals. [70, 87, 130, 109, 22]

Planning and reasoning for long-horizon agents is often studied through variants of guided decision making, search, and protocol-aware optimization. A common theme is that measured gains depend on how exploration is budgeted and how tool calls are scheduled relative to internal deliberation; as a result, work that reports improvements in efficiency, cost, or latency should be read jointly with the protocol constraints under which those improvements are achieved. This motivates distinguishing algorithmic changes in the planner from changes in the tool contract and execution environment. [138, 101, 30, 86, 151]

Memory and retrieval augmentations are frequently presented as interchangeable modules, but evaluation work suggests that retrieval policy and integration strategy materially affect both correctness and robustness. Systems that emphasize agent memory for tool use and browsing must confront citation and grounding issues, as well as the susceptibility of retrieval channels to injection and stale context. In practice, memory design also changes the agent loop: it determines what evidence can be used for planning and when the agent should re-query or revise. [103, 1, 64, 18, 77]

Learning-based adaptation and self-improvement papers extend agents beyond static prompting by introducing feedback-driven revision, preference shaping, and evaluation-guided tuning. These mechanisms make it easier to report improvements on benchmark suites, but they also increase the risk of overfitting to narrow protocols or exploiting metric artifacts. As a consequence, adaptation results are best interpreted together with the evaluation setting, including which tools are accessible, how many steps are allowed, and what failure behaviors are penalized. [106, 167, 61, 151]

Multi-agent coordination introduces additional protocol degrees of freedom: role definitions, communication formats, verification strategies, and aggregation rules. Related work covers both cooperative setups (task decomposition and planning) and adversarial or debate-style variants (verification and robustness), but comparisons are often inconsistent because messaging bandwidth, agent count, and stopping rules are not standardized. Empirical results therefore require careful reading of the interaction protocol and the evaluation harness rather than relying on high-level architectural labels alone. [25, 72, 73, 132]

Evaluation-focused papers and benchmark suites increasingly treat tool-using agents as a distinct class from static LLMs. They catalog task families and protocols, propose new suites with industrial workflows or realistic tool schemas, and emphasize reproducibility concerns such as leakage, hidden prompts, and uncontrolled environment variability. These contributions are complementary to architecture-focused surveys: they provide the protocol vocabulary (tasks, metrics, budgets, threat models) needed to make cross-paper synthesis interpretable. [83, 15, 144, 70, 87]

Security and governance work demonstrates that tool access expands the threat surface, and that standard success metrics can be misleading under attack. Analyses of prompt injection and tool abuse quantify attack success rates and privacy leakage, while guardrail and monitoring frameworks explore how to trade off robustness and benign utility under explicit adversarial models. These lines of work motivate incorporating threat models and safety constraints into the same protocol description used for performance evaluation, rather than treating security as a separate post-hoc concern. [150, 32, 162, 3, 53, 84, 82]

Recent work also treats agents as engineered systems, emphasizing execution sandboxes, tool-call datasets, and measurement harnesses that enable reproducible comparisons across models and interfaces. Examples include small-model function-calling pipelines and infrastructure-oriented evaluations, as well as system studies that surface how protocol choices (statefulness, time windows, interaction format) constrain external validity. Complementary system papers further stress that environment design and tracing affect reliability, from scaling interactive environments to specialized agent platforms and domain deployments. [23, 149, 58, 16, 115, 108, 107, 55, 29, 34, 24, 8, 119]

Taken together, the above literature suggests that a survey of LLM agents benefits from a protocol-aware synthesis: interface contracts and budgets determine the meaning of reported numbers, and risk models determine whether gains persist outside benign settings. The organization adopted here reflects this perspective by grouping work around (i) interfaces and loop design, (ii) core components for planning and memory, (iii) adaptation and coordination mechanisms, and (iv) evaluation and risks, so that comparisons are made within compatible protocols and limitations remain visible rather than being smoothed away. [137, 151, 83, 28, 150]

3 Foundations & Interfaces

Interfaces define what an agent can do, while the loop defines how reliably it does it. This chapter frames tool-using agents as closed-loop systems whose measured performance depends on action representations, tool schemas, and the execution boundary between the model and the environment. A practical consequence is that many apparent method differences collapse to interface and protocol differences once tool access, budgets, and observability are made explicit. [137] [70, 15]

The first subsection isolates the agent loop and action space: state representation, action granularity, recovery behavior, and how observations are fed back into decision making. The second subsection focuses on the tool contract itself - orchestration, permissions, sandboxing, and routing - because these choices determine both correctness and the dominant risk surface under deployment. Read together, the two subsections provide the ”execution semantics” needed to interpret later comparisons in planning, memory, and adaptation. [32, 84]

3.1 Agent loop and action spaces

Agent loops are where ”capability” becomes ”behavior”: they determine how a model turns an intent into a sequence of actions under a protocol. The central tension is that richer loops and action spaces can increase task success, but they also make results harder to verify because evaluation depends on hidden choices such as budgets, stopping rules, and what counts as a valid action or observation. A useful synthesis therefore treats the loop as an executable contract - not just a prompt style - and asks which parts of the contract are held fixed when numbers are reported. [137] [15, 144]

At a minimum, a tool-using agent loop instantiates a state -> decide -> act -> observe cycle, where ”act” includes structured tool calls and ”observe” includes tool outputs plus any environment feedback. The action space can be constrained (few tools with stable schemas) or expansive (many tools, open-ended arguments, dynamic tool discovery), and the choice affects both reliability and failure recovery. In practice, the same high-level loop label hides substantial variation in what is logged, what is remembered, and how errors are handled, which makes protocol-aware comparisons essential. Recent systems provide concrete instantiations of these loop and action-space choices under varied protocols. [55] [70, 152] [170, 168, 129, 81, 57, 112]

A first contrast is between loops that expand internal deliberation and loops that constrain external actions. More deliberation can reduce obvious mistakes, whereas tighter action representations (schemas, validators, or restricted tool sets) can reduce the probability of catastrophic tool misuse by narrowing what the agent can do. The literature increasingly mixes these approaches, but synthesis is clearest when the evaluation protocol states which constraints are enforced by the interface and which are learned or prompted behaviors. [152] [70, 16]

Benchmark landscapes already reflect this protocol dependence. One survey of agent evaluation reports 68 publicly available datasets spanning diverse tasks, which highlights how quickly evaluation has broadened without converging on a standard protocol vocabulary. At the same time, aggregate suites aim to compare agents across task families, but these comparisons remain brittle unless budgets and interaction formats are aligned. The practical takeaway is that ”agent loop” claims should be read together with the benchmark and the scoring rule, not in isolation. [15] [144, 16]

Concrete numbers illustrate why loop semantics matter. On two interactive decision-making benchmarks (ALFWorld and WebShop), ReAct-style reasoning-action prompting is reported to outperform imitation and reinforcement learning baselines by 34% and 10% absolute success rate, respectively, while using only one or two in-context examples. These gains are meaningful, but they are also protocol-specific: the action interface and termination criteria define what ”success” means and how much exploration is allowed. [137, 144]

Loop design is also inseparable from security because the action space creates an attack surface. For example, an analysis comparing function calling and MCP-style tool interfaces reports higher overall attack success rates for function calling (73.5% vs 62.59% for MCP), with different exposure patterns depending on whether the vulnerability is system-centric or model-centric. Such results imply that the same planner can appear robust or fragile depending on interface constraints and monitoring; in contrast, analyses that ignore the interface contract tend to attribute these differences to the planner or model alone. Loop comparisons should therefore include a threat model and the corresponding policy/sandbox setting. [32] [70, 24]

Efficiency and cost claims further amplify protocol sensitivity. Experiments on challenging agentic benchmarks such as GAIA and BrowseComp+ report that EvoRoute, when integrated into off-the-shelf agentic systems, can sustain or improve task performance while reducing execution cost by up to 80% and latency by over 70%. These improvements are best interpreted as properties of the full loop (routing policy + tool schedule + termination) under the benchmark's interaction format, rather than as model-only gains. [151, 15]

Training data and supervision choices can shift loop behavior even when the interface is fixed. One large-scale tool-agent dataset reports an average performance gain of about 20% over corresponding base models after supervised fine-tuning, improving results across coding, browsing, and tool-use benchmarks without domain-specific tuning. This suggests that the ”decide” step can be made more reliable via scale, but it also raises comparability questions: the resulting loop may call tools more or less often, changing cost and failure modes even when success rates improve. [109, 149]

Several limitations recur across loop evaluations and matter for how conclusions should transfer. Stateless agent designs can inflate apparent robustness by ignoring long-term drift, while short evaluation windows and single-run studies limit variance estimates and external validity. In addition, domain-specific agent deployments can exhibit strong gains on narrow protocols but unclear generality when tool access or environment dynamics change. These are not reasons to dismiss results; they are reasons to surface protocol fields explicitly so that later work can reproduce and stress-test the loop semantics. [115] [34, 29]

Taken together, the most stable way to compare agent loops is to treat the action space and protocol as the object of study: what actions are admissible, how observations are represented, what budgets apply, and what threats are modeled. This framing naturally motivates the next subsection on tool interfaces and orchestration, because the interface contract defines the "physics" of the action space that every loop - regardless of planner sophistication - must obey. [70] [151, 133]

If loop design determines which actions an agent can attempt, the tool/interface contract determines how those actions are encoded, constrained, and audited during execution.

3.2 Tool interfaces and orchestration

Tool interfaces are the boundary between a model's intent and executable actions, and orchestration is the policy that decides how to cross that boundary repeatedly. The practical trade-off is expressivity versus control: richer interfaces expand what an agent can attempt, but they also widen the space of failure modes and make behavior harder to constrain and verify. As a result, comparisons between tool-using agents are most interpretable when they treat the interface contract (schemas, permissions, observability) as part of the method rather than as an incidental engineering choice. [83] [137, 15]

Interface design starts with how actions are represented. Function-calling schemas and API specifications enable structured arguments and post-hoc validation, while natural-language tool descriptions favor flexibility but shift error detection into prompting or downstream checks. Orchestration then layers routing and control flow on top of these representations: selecting tools, handling retries, and deciding when to terminate under step and latency budgets. Even when two agents share the same base model, these interface and orchestration choices can dominate reliability. Recent systems explore these design choices across diverse tool contracts and budget regimes. [18] [68, 16] [27, 48, 36, 38, 127, 143]

A key contrast is between schema-first interfaces that make invalid actions unrepresentable and prompt-first interfaces that rely on the model to stay within an implicit contract. Schema-first designs can reduce argument errors and improve auditability, whereas prompt-first designs often require additional guardrails to prevent tool misuse and to make failures observable. In practice, hybrid approaches are common, but synthesis benefits from naming which parts of the contract are enforced by tooling (validators, sandboxes) and which are left to the model's behavior. [59] [102, 19]

Benchmarks increasingly encode this distinction by making the tool contract explicit. SOP-Bench, for example, operationalizes tool-use evaluation as end-to-end workflows: it contains over 1,800 tasks across 10 industrial domains, with APIs, tool interfaces, and human-validated test cases. This style of benchmark shifts attention from single-call correctness to orchestration reliability under realistic interface constraints. [87, 83]

Data resources further emphasize that tool interfaces are learnable behaviors when a stable contract exists. ToolMind introduces a large-scale tool-agent dataset with 160k synthetic instances generated over many tool types, aiming to improve tool selection and execution through supervised training. Such datasets can raise benchmark performance, but they also increase the importance of protocol reporting: tool availability, schema versions, and execution environment details can change what the model is actually learning to do. [130, 16]

Orchestration methods can be viewed as policies over a tool graph: they decide which tool to call next, how to sequence tools, and how to recover from partial failures. Automation-oriented systems focus on discovering tools, composing them, and learning routing heuristics; in contrast, evaluation-oriented systems focus on measuring tool correctness, failure recovery, and budget sensitivity under fixed protocols. This distinction matters because an orchestration improvement that reduces calls under one harness can look like a planning improvement even when it is primarily a routing or caching change. [52] [13, 31]

The evaluation literature reinforces that interface details must be represented in the protocol, not inferred from narrative descriptions. Surveys and core benchmark studies propose taxonomies for agent evaluation, but they also expose how heterogeneous current reporting is, especially for tool access assumptions and budget constraints. A protocol-aware perspective therefore treats tool interface fields - schemas, permissions, sandboxing, observability - as the minimal metadata required to make cross-paper comparisons meaningful. [83] [79, 15, 16]

Tool interfaces also interact with coordination and state: orchestration is not only about single-agent routing but also about managing shared resources, memory, and multi-agent communication under constraints. For instance, memory-augmented tooling and coordination frameworks often change the effective action space by adding ”retrieve” or ”delegate” as first-class operations, which can improve task completion but can also obscure where errors originate (model reasoning, tool outputs, or orchestration policy). [73, 68]

Several limitations recur in interface and orchestration research. Expressive tool contracts can amplify prompt injection and tool-side manipulation if permissions and monitoring are weak, while overly restrictive contracts can reduce capability by preventing the agent from taking corrective actions. Moreover, benchmarks may underrepresent long-tail tool failures (timeouts, partial outputs, schema drift), so reported gains can be sensitive to the specific harness design. These caveats motivate evaluation protocols that stress tool realism and that report interface fields as first-class variables. [164] [26, 19, 59]

In practice, tool interfaces and orchestration define the operational semantics that later components - planners, memory modules, and adaptation mechanisms - must work within. Once the interface contract is fixed, planning and memory choices become easier to compare because the action space and observation format are stable; when the contract shifts, many ”algorithmic” claims reduce to protocol differences. This motivates the next chapter, which treats planning and memory as components whose behavior is only interpretable under explicit interface and budget assumptions. [137, 83]

4 Core Components (Planning + Memory)

Once interfaces define the action space, the next bottleneck is how agents choose actions over time under uncertainty and budget constraints. This chapter treats planning and memory as protocol-sensitive components: reported gains often depend as much on search breadth, tool-call scheduling, and termination rules as on the underlying model. Efficient variants therefore need to be interpreted jointly with the evaluation protocol (task format, step limits, latency/cost budgets) used to measure them. [138] [151, 83]

The planning subsection emphasizes control loops and deliberation: when agents branch, how they score alternatives, and how plans are revised after tool feedback. The memory subsection focuses on what evidence the agent can condition on - retrieval policy, write/update rules, and how retrieved context is verified - because memory modules can shift both correctness and robustness without changing the nominal planner. Together, these components explain why ”stronger reasoning” claims frequently hinge on grounded state and comparable protocols rather than prompt wording alone. [103] [1, 64]

4.1 Planning and reasoning loops

Planning and reasoning loops determine how an agent turns observations into a sequence of actions under uncertainty. The recurring tension is deliberation depth versus cost: deeper planning can improve reliability, but it increases latency, step budgets, and sensitivity to protocol choices such as termination and tool access. As a result, planning papers are best compared as control-loop designs evaluated under explicit budgets, rather than as purely ”better reasoning” claims detached from the execution harness. [42] [165, 39]

At a high level, planning loops instantiate a control architecture (planner, executor, critic, verifier) and decide when to branch, when to commit, and how to revise after tool feedback. Some approaches emphasize lightweight iteration with frequent tool calls, while others allocate more budget to internal deliberation before acting. These design choices change not only task success but also the distribution of failures (wrong tool choice, brittle step ordering, or compounding errors), which motivates protocol-aware reporting of step limits, retry rules, and observation formats. Recent planning systems instantiate different trade-offs between deliberation, tool-call scheduling, and revision under constraint. [17] [5, 93] [44, 94, 104, 157, 56, 159]

A useful contrast is between reactive loops that act early and revise often, and planning-heavy loops that explore alternatives before committing. Reactive loops can be efficient when tools provide strong signals, whereas planning-heavy loops can be more robust when early actions are costly or irreversible. However, the two families can look identical in aggregate success metrics if their budgets differ; comparisons therefore need to normalize for latency/cost and to report how much of the budget is spent on deliberation versus tool execution. [137] [14, 30]

Concrete evaluations illustrate how domain constraints shape planning design. One penetration-testing agent is evaluated using three LLMs (Llama-3-8B, Gemini-1.5, and GPT-4) and is applied to navigate 10 target websites, making the interaction protocol (what is observable, what actions are permitted) central to interpreting its reported effectiveness. In such settings, planning quality is inseparable from the environment interface and from how failures are detected and recovered. [86, 165]

Efficiency-oriented results further highlight budget sensitivity. In one study, agents trained with the proposed method show more efficient tool use, with inference speed reported to be about 1.4x faster than baseline tool-augmented LLMs on the evaluated tasks. These gains are valuable for deployment, but they should be read as properties of the full planning loop under a specific harness, because changes in tool-call scheduling or caching can dominate latency without changing the underlying model. [30, 42]

Benchmark suites that span multiple environments help stress-test whether planning behaviors transfer. AgentSquare reports experiments across six benchmarks covering web, embodied, tool-use, and game scenarios, which makes protocol alignment (task format, scoring, budgets) a prerequisite for synthesis. Multi-benchmark reporting is an improvement over single-task demos, yet head-to-head comparison remains fragile when different papers implicitly use different termination rules or tool access assumptions. [101, 47]

Realistic evaluation targets also motivate planning loops that can handle hard, tool-rich tasks. MCP-Universe is introduced as a benchmark designed to evaluate LLMs in realistic and difficult settings, shifting attention toward robust planning under complex tool interfaces rather than narrow prompt-chasing. Such benchmarks make it easier to identify which planning design choices are stable under protocol variation, because they force explicit specification of tools, observations, and failure recovery. [76, 163]

Testing and measurement methodology matters because planning loops can overfit to interaction quirks. Work on agent testing and diagnostic benchmarks emphasizes that a single aggregate score can hide failure modes such as brittle tool sequencing, hidden leakage, or reliance on privileged hints. Protocol-aware evaluation therefore benefits from structured test generation, ablations that isolate planning components, and reporting that separates model improvements from orchestration artifacts. [50] [92, 85]

Several limitations recur across planning papers. Planning-heavy loops can degrade when observations are noisy or adversarial, while reactive loops can fail catastrophically when early actions have high downside. Moreover, planning comparisons are often under-identified: if budgets, tool availability, and retry policies are not matched, it is unclear whether reported gains reflect better control logic or simply more compute. These caveats motivate explicit protocol fields and controlled comparisons, especially when claims target general agent capability rather than a single benchmark. [47] [147, 160]

In sum, planning and reasoning loops are best synthesized as control policies operating under a protocol: they trade off reliability, cost, and robustness depending on how much evidence is available and how expensive actions are. This perspective naturally connects to memory and retrieval, because planning quality depends on what state and evidence are available for decision making; the next subsection therefore treats memory as a component that changes what planning can reliably condition on. [65] [124, 101]

Planning quality depends on what state and evidence are available at decision time, so memory and retrieval become the mechanism that makes long-horizon deliberation reliable under budget.

4.2 Memory and retrieval (RAG)

Memory and retrieval determine what evidence an agent can condition on and how long-horizon state is represented across steps. The core tension is persistence versus freshness: retaining more context can improve performance on multi-step tasks, but it also raises staleness, contamination, and verification challenges. Because retrieval is itself an action in the loop, memory design must be compared under explicit protocols - what can be retrieved, how often, and how retrieved content is validated - rather than being treated as a plug-in component. [103] [46, 136]

In tool-using agents, memory spans several layers: short-term scratchpads, episodic traces of prior actions, and long-term stores indexed by text, structure, or learned embeddings. Retrieval-augmented generation (RAG) is a common mechanism, but its behavior depends on indexing, query formation, and integration strategy (constraints vs suggestions vs authoritative facts). These choices shape both correctness and robustness, especially when tools can write to or read from the same memory substrate. Recent memory designs illustrate a wide range of retrieval policies and memory substrates, which is why validation and write governance matter. [118] [90, 45] [169, 120, 148, 60, 123, 67]

A useful contrast is between retrieval-as-context and retrieval-as-control. Retrieval-as-context treats retrieved content as additional conditioning information, whereas retrieval-as-control uses retrieved items to constrain action selection (e.g., which tool to call or which plan branch to pursue). However, the two can be indistinguishable in aggregate scores if verification is weak; protocol-aware evaluation should therefore report how retrieval is validated (citation checks, self-consistency, or external verification) and how memory writes are governed. [146] [154, 55]

Concrete benchmark design makes these distinctions measurable. MuaLLM, for example, introduces two custom datasets: RAG-250, targeting retrieval and citation performance, and Reasoning-100, focused on multistep reasoning. This formulation clarifies what the memory module is expected to provide (relevant evidence and grounded citations) and what counts as a failure mode (hallucinated support, stale context, or brittle query behavior). [1, 103]

Compression-oriented memory systems highlight a different trade-off: retaining more context often requires summarization or distillation. Meta-RAG reports condensing codebases by an average of 79.8%, which can make long contexts tractable for agents operating over large repositories. Yet compression changes the evidence surface: in contrast to raw retrieval, summaries can introduce abstraction errors that are hard to detect without explicit verification steps, so evaluation protocols should separate retrieval quality from summarization fidelity. [111, 46]

Multi-benchmark evaluations also stress that memory benefits are domain-dependent. AgentSwift is evaluated across seven benchmarks spanning embodied, math, web, tool, and game domains, and reports average gains across this diverse suite. Such breadth improves external validity, but synthesis still requires protocol alignment: different benchmarks expose different observation noise and different opportunities for retrieval to help or harm. [64, 101]

Retrieval policy is increasingly treated as an object of study rather than a default setting. Survey and analysis work emphasizes that retrieval frequency, query selection, and ranking criteria can materially change agent behavior under the same planner. As a result, comparisons between memory-augmented agents should report retrieval budgets and the interface between retrieval and tool calls (e.g., whether retrieval results are treated as tool outputs with their own trust and permission model). [63] [136, 18]

Memory also interacts with planning: planning quality depends on what state is reliably available at decision time, and memory determines which evidence is accessible under a budget. Systems that combine planning loops with retrieval can appear stronger simply because they re-query more often, whereas systems that write structured traces can reduce rework at the cost of accumulating stale assumptions. Protocol-aware evaluation therefore benefits from tracking memory write policies and from measuring how often plans are revised due to retrieved evidence. [42] [140, 141]

Several limitations and risks recur in memory-based agents. Retrieval channels can carry adversarial content or prompt injections, and long-lived memories can encode false premises that persist across tasks. In addition, some benchmarks under-specify what counts as permissible evidence, making it unclear whether retrieval gains reflect better grounding or leakage through the environment. These issues motivate explicit threat models and verification steps for memory, especially in settings where memory can influence tool invocation. [77] [55, 45]

More broadly, memory and retrieval are most useful when treated as protocolized components: they expose a budgeted evidence interface to the planner and must be evaluated for both correctness and robustness. This framing connects naturally to learning and adaptation, because feedback loops often operate on the same memory traces and can amplify both improvements and failures over time. [90] [21, 80]

5 Learning, Adaptation & Coordination

Adaptation mechanisms change the agent over time, while coordination mechanisms change the effective agent by distributing cognition across roles. This chapter organizes learning and multi-agent systems around the feedback signals and interaction protocols they assume: what counts as success, how errors are detected, how revisions are applied, and how multiple agents reconcile disagreements. These assumptions directly shape both benchmark outcomes and failure modes such as reward hacking, overfitting to narrow protocols, and brittle aggregation. [106] [167, 83]

The self-improvement subsection examines critique and revision loops and their stability under repeated interaction, focusing on what is learned (prompts, policies, tool usage) and under which evaluation constraints. The multi-agent subsection covers role specialization and communication/aggregation rules, highlighting that coordination gains are inseparable from protocol choices such as message bandwidth, agent count, and stopping conditions. Read together, they motivate reporting feedback and coordination protocols as first-class metadata when claiming generality. [25] [72, 73]

5.1 Self-improvement and adaptation

Self-improvement makes agents dynamic systems: they change their behavior based on feedback, experience, or evaluation signals. A core trade-off is adaptability versus stability - systems that revise themselves can improve over time, but they also risk drifting, overfitting to narrow protocols, or becoming harder to evaluate and control. A protocol-aware synthesis therefore asks not only whether performance increases, but also what is being updated (prompts, policies, tool usage) and under which constraints the update is valid. [167] [21, 117]

Adaptation mechanisms span several families. Reflection and critique loops revise intermediate reasoning or tool plans after failures; preference- or reward-driven methods shape behavior via feedback signals; and evaluation-driven tuning uses benchmark performance as an optimization target. These approaches differ in how tightly they bind to an evaluation harness: some treat the harness as a learning environment, whereas others treat it as a diagnostic that informs manual or automated revisions. Recent systems operationalize adaptation via different feedback signals and update targets, making constraints part of the claim. [10] [7, 39] [145, 33, 131, 97, 135, 156]

A useful contrast is between updates that are explicitly constrained by a protocol and updates that implicitly absorb protocol quirks. Constrained updates can improve reliability under matched tool access and budgets, whereas unconstrained updates may pick up brittle shortcuts that do not transfer when tools, prompts, or termination rules change. In practice, many adaptation results should be interpreted as conditional on the training/evaluation setting, and reporting should make those conditions explicit to avoid overclaiming. [139] [62, 144]

Concrete evaluations show that adaptation can deliver large gains on tool-use benchmarks. On two multi-turn tool-use agent benchmarks (M3ToolEval and TauBench), the Self-Challenging framework reports over a two-fold improvement, illustrating that feedback and revision policies can be as important as base-model capability for interactive success. These results also highlight why protocol fields matter: the same revision policy can trade off cost, latency, and robustness depending on how tool calls and retries are counted. [167, 35]

Benchmark coverage has expanded rapidly, and this growth changes what "improvement" means. A comprehensive survey connects 50+ benchmarks to solution strategies, underscoring that adaptation can be measured across heterogeneous task families with different scoring rules and interaction formats. This heterogeneity motivates a conservative stance: gains on a single suite should be treated as evidence about that suite's protocol, not as a blanket statement about agent capability. [35] [144, 21]

Foundational agent-loop results also motivate adaptation as a response to long-horizon brittleness. ReAct-style agents show large success-rate gains on interactive tasks (e.g., ALFWorld and WebShop), but they still exhibit failure modes such as premature commitment and error compounding. Adaptation mechanisms can be viewed as attempts to correct these failure modes online, yet the magnitude of benefit depends on whether the evaluation protocol rewards cautious verification or merely short-horizon completion. [137, 167]

Data and tooling can act as implicit adaptation channels by changing what the agent learns to do with tools. Tool-use training resources and structured supervision can improve tool selection and execution, but they can also shift behavior toward the idiosyncrasies of a particular tool contract or benchmark harness. In contrast to pure prompting, training-based adaptation should therefore report the tool definitions, access constraints, and the distribution of tasks used for supervision. [22] [62, 90]

Several method lines explore how to make adaptation more stable and controllable. Some approaches emphasize bounded revision policies and explicit failure triggers, while others focus on iterative improvement over longer horizons or on domain-specific adaptation where evaluation is more structured. These designs often trade off sample efficiency and reliability: aggressive updates can improve quickly but may destabilize behavior, whereas conservative updates may be slower but easier to audit. [166] [126, 128, 41]

Limitations and risks are central for self-improving agents. Adaptation can exploit metric artifacts, reduce interpretability of the decision policy, or create reward-hacking behaviors when feedback signals are misaligned. In addition, ethical and governance constraints can matter in practice: when agents are optimized for performance, it becomes unclear how to bound behavior in sensitive settings without explicit constraints and monitoring. These caveats argue for reporting stability diagnostics and for stress-testing adaptation under shifts in tools, prompts, and threat models. [113] [21, 3]

A practical implication is that self-improvement should be evaluated as a controlled process: what feedback is used, what is updated, and how improvements trade off robustness, cost, and safety under the stated protocol. This framing also sets up multi-agent coordination, where adaptation can be distributed across roles and where stability depends on communication and aggregation rules as much as on individual agent learning. [61] [117, 151]

Once behavior can be updated from feedback, the same stability question reappears at the team level, where coordination protocols determine how roles share evidence, verify claims, and avoid correlated drift.

5.2 Multi-agent coordination

Multi-agent coordination treats an ”agent” as a team: capability comes from specialization and verification across roles rather than from a single loop. A central trade-off is specialization versus coordination - dividing labor can boost performance, but it adds communication overhead, introduces new failure modes, and makes evaluation protocol choices (agent count, message budget, stopping rules) decisive. A protocol-aware synthesis therefore focuses on interaction contracts: what roles exist, how information is exchanged, and how disagreements are resolved. [132] [98, 142]

Coordination patterns span several families. Some systems use role specialization for decomposition (planner, executor, reviewer), others use debate or referee mechanisms for verification, and others use aggregation (vote, rank, consensus) to stabilize noisy decisions. These patterns differ in what they assume about observability and trust: coordination can amplify errors if agents share the same flawed evidence, whereas diversity can help when independent perspectives are available. Recent multi-agent systems instantiate these coordination patterns across roles, message budgets, and aggregation rules. [9] [88, 153] [74, 11, 105, 49, 20, 145]

A useful contrast is between centralized orchestration and decentralized coordination. Centralized orchestration can impose consistent tool access and enforce budgets, whereas decentralized setups may be more flexible but can suffer from runaway communication and unstable convergence. However, the two are often compared under different protocols; careful evaluation therefore needs to report message budgets, aggregation rules, and whether agents share memory or tool outputs. [102] [125, 142]

Concrete benchmarks show how sensitive coordination is to evaluation setup. Using only 400 problem instances from the Berkeley Function-Calling Leaderboard (BFCL), one method reports competitive in-distribution performance while improving generalization, highlighting that coordination benefits can be measured even under relatively small test sets. At the same time, small test sets can under-sample rare coordination failures, so reported gains should be interpreted with attention to variance and protocol details. [72, 142]

Memory-sharing and tool mediation are common coordination mechanisms. MemTool evaluates different coordination modes across 13+ LLMs on the ScaleMCP benchmark, with experiments over 100 consecutive user interactions; this style of evaluation makes coordination overhead observable and ties performance to an explicit interaction protocol. Such results underscore that coordination is not free: policies that improve success can increase cost or latency, which must be part of the claim when coordination is justified as an engineering strategy. [73, 31]

Coordination is also used to improve planning and tool-integrated reasoning on hard agent benchmarks. GiGPO is evaluated on ALFWorld and WebShop as well as tool-integrated reasoning on search-augmented QA tasks, using Qwen2.5-1.5B/3B/7B-Instruct as base models. This illustrates a common pattern: multi-agent designs often trade off model size against interaction structure, using coordination to compensate for weaker individual agents under a fixed tool interface. [25, 72]

Evaluation methodology matters because multi-agent systems can ”win” by exploiting protocol artifacts (e.g., hidden hints in shared memory, or aggregation rules that mask individual failures). Work on generating ground truth and validating evaluation metrics emphasizes that agent scoring must be tied to observable behaviors, not only to end outcomes, especially when multiple agents contribute. Protocol-aware evaluation can therefore benefit from logging communication traces and attributing errors to specific roles or messages. [149, 142]

Several coordination designs emphasize robustness through verification, but robustness depends on the threat model. Coordination can reduce single-agent hallucinations, yet it can also create correlated failures when agents share the same adversarial context or when a single compromised tool output is broadcast to the team. In contrast to single-agent settings, coordination protocols must specify trust boundaries: what is verified, who can call tools, and how tool outputs are sanitized before being shared. [153] [102, 77]

Limitations and risks are prominent. Coordination increases surface area for instability (non-convergence, oscillation, or premature consensus), and it can degrade under tight latency or message budgets. Moreover, coordination success can be brittle to agent count and role definitions, making it unclear whether reported gains reflect general mechanisms or narrow protocol tuning. These caveats motivate reporting coordination protocol fields as explicitly as tool schemas. [98] [77, 66]

One way to read the evidence is that multi-agent coordination should be synthesized as a family of protocols, not as a single architectural choice: the same roles can behave very differently under different budgets, memory-sharing policies, and aggregation rules. This perspective connects directly to evaluation and risk considerations, where the choice of benchmarks, logging, and threat assumptions determines which coordination claims are reproducible and which are artifacts of a particular harness. [142] [98, 133]

6 Evaluation & Risks

Evaluation determines which claims about agents are meaningful, and risk models determine which gains survive outside benign settings. This chapter connects benchmark design, protocol fields, and threat models so that reported numbers can be interpreted as statements about closed-loop systems rather than isolated model snapshots. In practice, differences in tool access, budgets, hidden prompts, and environment variability can dominate conclusions unless they are controlled or at least explicitly reported. [83] [15, 28]

The benchmarks subsection surveys task suites and protocol conventions for tool-use and long-horizon evaluation, emphasizing how metrics interact with interaction formats and budgets. The risks subsection focuses on security and governance: prompt injection, tool abuse, and monitoring/guardrail strategies that trade off robustness with task utility under explicit adversarial assumptions. Together, they provide a protocol-aware lens for interpreting both performance improvements and safety claims in the rest of the paper. [150] [32, 84]

6.1 Benchmarks and evaluation protocols

Benchmarks and evaluation protocols decide which claims about agents are meaningful. A core tension is coverage versus comparability: broader suites capture more behaviors, but they also make head-to-head comparison fragile when protocols and constraints differ. A protocol-aware view therefore treats evaluation as a specification problem - tasks, metrics, budgets, tool access, and threat assumptions must be explicit - rather than as a post-hoc score attached to a system description. [83] [79, 51]

Evaluation protocols for tool-using agents include more moving parts than static LLM evaluation. Beyond task and metric choice, protocols define interaction formats (observations, action schemas), budget constraints (steps, latency, cost), termination rules, and what constitutes permissible evidence. Human evaluation and safety constraints can further change the interpretation of the same success rate. As a result, two papers can report similar scores while evaluating different objects, motivating explicit protocol fields in benchmark metadata and in paper reporting. Recent benchmark work illustrates how these protocol fields are being made explicit for comparability. [83] [114, 122] [12, 69, 110, 116, 4, 155]

A useful contrast is between benchmark suites designed for breadth and benchmarks designed for diagnostic comparability. Broad suites aim to cover diverse environments and task families, whereas diagnostic benchmarks isolate specific capabilities such as tool invocation correctness, grounded retrieval, or long-horizon planning under a fixed interface. In practice, both are necessary: breadth supports external validity, but diagnostic structure is needed to attribute gains to specific components rather than to protocol quirks. [89] [161, 58]

Surveys and taxonomies of agent evaluation increasingly emphasize this protocol gap. They catalog benchmark dimensions and propose organizing frameworks, but they also highlight that many evaluation reports omit key fields such as tool access, budget assumptions, or whether hidden prompts and environment variability are controlled. A direct implication is that evaluation results should be treated as conditional evidence unless protocol metadata is sufficient for replication. [83] [51, 54]

The benchmark landscape is also expanding rapidly. One comprehensive survey connects 50+ benchmarks to corresponding solution strategies, which helps readers map what is being tested and which methods are commonly applied. At the same time, this growth makes it easier to cherry-pick results; protocol-aware synthesis therefore benefits from tracking which benchmarks share the same interaction format and constraints, and which do not. [35, 89]

Tool-use agent benchmarks illustrate how interaction constraints shape measured capability. On multi-turn tool-use benchmarks such as M3ToolEval and TauBench, self-improvement frameworks report large gains, suggesting that revision policies and tool-selection behaviors are central. However, these benchmarks also depend on how tool calls are counted, what constitutes a valid argument, and how partial failures are handled; without those protocol details, it is unclear whether gains reflect better decision making or looser harness assumptions. [167, 79]

Security-oriented benchmarks make the protocol and threat model explicit by construction. On the AgentDojo prompt injection benchmark, RTBAS reports preventing all targeted attacks with only a 2% loss of task utility under attack, while other studies quantify that LLM utility can drop by 15-50 percentage points under attack with average attack success rates around 20% in similar settings. These results show why evaluation protocols must include adversary models and defense policies; otherwise, robustness claims are not comparable. [162] [3, 37]

Measurement methodology is therefore part of the benchmark contribution. Work on core evaluation design and on measuring agent behavior emphasizes logging and attribution: what the agent saw, what it did, and how errors map to protocol fields. This is especially important for tool-using agents because end outcomes can hide brittle behaviors such as over-reliance on privileged tool outputs, hidden leakage, or unstable retries. Benchmark suites that publish traces and explicit schemas make it easier to diagnose such issues. [79] [114, 58, 122]

Limitations remain. Benchmarks can leak information through environment artifacts, prompt templates, or tool descriptions, and many suites underrepresent long-tail failures such as timeouts and partial tool outputs. In addition, comparability can break when different papers use different tool schemas or evaluation scripts under the same benchmark name. These caveats motivate reproducible releases (schemas, scripts, and traces) and caution against overinterpreting small score differences. [28] [45, 19]

Taken together, benchmarks and protocols should be read as defining the object of evaluation: a closed-loop system under constraints. Protocol-aware synthesis therefore emphasizes which fields are comparable across papers and which require normalization before claims can be aggregated. This perspective also sets up the next subsection on risks, where threat models, monitoring, and governance constraints become part of the evaluation protocol rather than external concerns. [83] [28, 2]

Protocol metadata makes comparisons meaningful, but deployment decisions additionally hinge on threat models and governance controls that define what robustness means for tool-using systems.

6.2 Safety, security, and governance

Safety and governance become first-order concerns once agents can act through tools: capability increases utility, but it also widens the attack surface and raises containment requirements. The key tension is therefore capability versus safety - stronger actions can complete harder tasks, yet they can also enable prompt injection, tool abuse, and data exfiltration when interfaces and monitoring are weak. A protocol-aware synthesis treats security as part of the evaluation contract: threat models, permissions, and logging determine what robustness claims mean. [150] [28, 121]

Threat models for tool-using agents include prompt injection (in user inputs, tool descriptions, or retrieved content), malicious tool outputs, and policy bypass via argument manipulation. Governance mechanisms span guardrails, monitors, sandboxing, and auditing. These controls are tightly coupled to the tool interface: if tool schemas are underspecified or permissions are broad, agents can be coerced into actions that look like "task completion" but violate safety objectives. Recent governance and safety work spans a range of threat models and control surfaces, so robustness claims should be read as contract-dependent. [24] [59, 45] [158, 100, 95, 75, 40, 43]

Empirical security analyses highlight how interface choice affects robustness. One comparison of function calling and MCP-style tool interfaces reports higher overall attack success rates for function calling (73.5% vs 62.59% for MCP), with different exposure patterns depending on whether the vulnerability is system-centric or model-centric. This implies that robustness cannot be inferred from model quality alone; in contrast, security claims must be interpreted as properties of a full system under a specific interface contract and monitoring policy. [32, 59]

Guardrail frameworks attempt to reduce harmful tool use while preserving benign utility. TS-Flow, for example, reports reducing harmful tool invocations of ReAct-style agents by about 65% on average and improving benign task completion by roughly 10% under prompt injection attacks. These results illustrate a common trade-off: stronger defenses can alter the agent loop and therefore change what success metrics measure, which motivates reporting both robustness and utility under matched protocols. [84, 165]

Taxonomies of attacks help make the risk surface explicit and comparable. MSB, for instance, catalogs attacks such as name collision, preference manipulation, prompt injections embedded in tool descriptions, out-of-scope parameter requests, user-impersonating responses, and mixed-attack combinations. Such taxonomies are useful because they connect failure modes to interface and governance fields (tool descriptions, schema validation, permission models) that can be controlled and audited. [150, 121]

Evaluation suites increasingly incorporate these threat models into benchmarks. RAS-Eval reports 80 test cases and 3,802 attack tasks mapped to 11 CWE categories, with tools implemented in JSON, LangGraph, and MCP formats, enabling controlled comparisons across interface variants. Attack suites also quantify that tool-use scenarios can sustain very high attack success rates and privacy leakage with little impact on benign-task execution, which underscores why success metrics alone are insufficient. [28, 82]

Monitoring and red-teaming workflows provide another layer of governance. Work on monitor red teaming emphasizes varying monitor awareness, adversarial strategies to evade detection, and datasets/environments designed to stress safety mechanisms under realistic constraints. Such approaches treat robustness as an interaction between the agent and the monitor policy, so evaluation protocols should report both sides of the system to avoid attributing outcomes solely to the agent model. [53] [78, 71]

Containment and sandboxing mechanisms aim to limit blast radius even when an agent is compromised. Two-way sandboxing designs and secure execution frameworks propose isolating both the agent from the host and the host from the agent, which becomes important when agents can call arbitrary tools or execute code. These system-level controls interact with orchestration: restrictive sandboxes can prevent harmful actions; however, they can also break benign workflows unless interfaces and policies are designed jointly. [134] [96, 23]

Several limitations remain for safety evaluations. Threat models are often incomplete, and defenses can be brittle to distribution shifts in tool descriptions, retrieval content, or user behavior. In addition, system studies report that even strong models can exhibit significant performance limitations on realistic tasks, which complicates the interpretation of robustness claims when failures may stem from capability gaps rather than adversarial pressure alone. These caveats motivate combined reporting of capability, robustness, and protocol fields. [76] [8, 37]

In sum, safety, security, and governance for tool-using agents is best framed as protocol design: interfaces, permissions, monitoring, and auditing jointly define what the agent is allowed to do and how failures are detected. This framing connects back to the benchmark discussion: without threat models and governance fields, published numbers are not comparable and do not support deployment decisions. A practical next step is to standardize reporting of tool contracts, sandbox settings, and attack suites as part of benchmark metadata and system papers. [121] [28, 6]

7 Discussion

Across agent systems, protocol assumptions - tool access, budgets, termination, and threat models - are the dominant hidden variables that decide whether two reported results are comparable. A practical implication is that architectural labels (planner, memory, multi-agent) are insufficient without an explicit interface contract: two planners behave differently when tool schemas are underspecified, tool outputs are noisy, or execution is not sandboxed. This suggests that future benchmarks should publish protocol fields as first-class metadata and treat interface variability as a controlled factor rather than an uncontrolled nuisance. [83, 70, 28]

Another cross-cutting observation is that efficiency claims (lower cost, fewer steps, reduced latency) are increasingly used as a proxy for general progress, yet they are especially sensitive to protocol details. Systems that optimize routing or execution schedules can deliver large gains under a fixed harness, but the same techniques may shift failure modes when tool availability, environment variability, or monitoring changes. A useful next step is to standardize reporting of budgets and to separate model-level improvements from orchestration-level improvements so that progress can be attributed. [151, 15, 32]

Security results underline why evaluation and deployment cannot be separated for tool-using agents. Prompt injection and tool abuse demonstrate that benign-task performance can coexist with severe failures in confidentiality or integrity, and defenses often trade off robustness with task utility. Rather than treating safety as an add-on, protocol-aware evaluation can incorporate threat models and monitoring constraints into benchmark design, making robustness claims interpretable and reproducible. [150, 162, 84, 53]

Finally, evidence quality remains uneven across sub-areas: some results are supported by broad benchmark coverage, while others rely on narrow task suites or single-run case studies. A conservative synthesis therefore prioritizes comparisons that are stable under protocol variation and highlights where stronger verification is needed (e.g., full-text protocol details, tool definitions, and ablations that isolate interface effects). [15, 144, 115]

8 Conclusion

Tool-using LLM agents are best understood as closed-loop systems whose behavior depends jointly on the agent loop, the interface contract, and protocol constraints such as budgets and threat models. This perspective clarifies why results can be difficult to compare across papers:

changing tool schemas, permissions, or termination rules can shift measured performance as much as changing the underlying model or planner. [137, 70, 83]

Three takeaways follow. First, interface contracts determine what evaluation claims are even interpretable, so benchmarks and papers should publish tool schemas and execution constraints alongside metrics. Second, planning, memory, and adaptation mechanisms should be compared within compatible protocols, with explicit accounting for cost and latency when those are part of the claim. Third, safety and governance constraints must be integrated into evaluation because tool access expands the attack surface and creates new failure modes that benign metrics can hide. [28, 151, 150, 84]

The most actionable path forward is to standardize protocol reporting and benchmark metadata so that future work can separate model-level capability from orchestration choices and can evaluate robustness under explicit threat models. [15, 83, 53]

A Tables

Appendix Table A1. Agent loop patterns and interface assumptions (representative works).

Pattern	What changes in the agent loop	Interface / assumptions (what must be specified)	Key refs
Think-Act-Observe prompting	Interleave reasoning and tool actions; update plans from observations	Tool schema + observation format; step budget/state	[137]
Self-supervised tool calling	Model learns to insert tool calls during generation	Tool execution sandbox; consistent tool outputs	[99]
Reflection-based self-improvement	Use self-critique/feedback to revise prompts or plans after failures	Feedback signal; memory of past attempts	[106]
Search/planning over thoughts	Explore candidate reasoning branches before acting	Scoring/selection rule; budgeted search	[138]
Skill/library-driven embodied agent	Acquire reusable skills/tools for long-horizon tasks	Environment API; skill representation + retrieval	[119]
Dataset-driven tool-use SFT	Scale instruction/traces to improve tool selection and execution	Standardized tool definitions; train/eval harness	[130, 109, 22]
Protocolized tool interface (MCP / function calling)	Standardize discovery, invocation, and tool-side metadata	Schema/versioning; permissions/sandbox; observability	[70, 32]
Multi-agent coordination	Split subtasks across agents; aggregate via vote/referee	Role protocol; message format; conflict resolution	[25, 132]
Cost/latency-aware routing	Optimize tool routing to reduce cost/latency without losing task success	Budget metrics + tracing; cost-aware policy	[151]
Safety/guardrail layer	Detect and mitigate prompt injection and malicious tool outputs	Threat model; safe tool invocation policy	[150, 84, 28]

Appendix Table A2. Benchmarks and evaluation settings for tool-using agents (examples).

Benchmark / setting	What it tests	Typical metric(s)	Notes / constraints	Key refs
ALFWorld	Interactive, multi-step decision making	Success rate / completion	Environment interaction; step budget	[137]
WebShop	Web navigation + tool-augmented reasoning	Success rate / task completion	Long-horizon browsing; noisy observations	[137]
GAIA	General agentic problem solving	Task success	Often paired with cost/latency analysis	[151]
BrowseComp+	Web browsing competency	Task success; cost/latency	Tool calls; latency-sensitive workflows	[151]
MCPAgentBench	Tool-use via MCP definitions	Pass rate / tool correctness	Realistic tool schemas/protocols	[70]
SOP-Bench	Industrial API workflows	Task success on 1,800+ tasks	Domain diversity; human-validated cases	[87]
BFCL (function calling)	Function-call selection + arguments	Leaderboard score / success	Focus on structured tool calls	[72]
M3ToolEval, TauBench	Multi-turn tool-use agents	Benchmark score / success	Emphasizes multi-step tool use	[167]
MMAU	Broad agent capabilities suite	Aggregate score	Multi-task prompting protocol	[144]
AgentDojo (prompt injection)	Robustness to injection attacks	ASR vs utility	Defense trade-offs are common	[162, 3]
RAS-Eval and related attack suites	Tool-use security evaluation	ASR / privacy leakage	Attack-task suites; multiple tool formats	[28, 82]
Tool-safety guardrail evaluation	Harm reduction under attack	Harmful tool invocations; utility	Requires explicit threat model + benign baseline	[84]

References

[1] Pravallika Abbineni, Saoud Aldowaish, Colin Liechty, Soroosh Noorzad, Ali Ghazizadeh, and Morteza Fayazi. Muallm: A multimodal large language model agent for circuit design assistance with hybrid contextual retrieval-augmented generation. *arXiv preprint arXiv:2508.08137v1*, 2025. URL http://arxiv.org/abs/2508.08137v1.

[2] Kushal Agrawal, Frank Xiao, Guido Bergman, and Asa Cooper Stickland. Why do language model agents whistleblow? *arXiv preprint arXiv:2511.17085v2*, 2025. URL http://arxiv.org/abs/2511.17085v2.

[3] Meysam Alizadeh, Zeynab Samei, Daria Stetsenko, and Fabrizio Gilardi. Simple prompt injection attacks can leak personal data observed by llm agents during task execution. *arXiv preprint arXiv:2506.01055v1*, 2025. URL http://arxiv.org/abs/2506.01055v1.

[4] Thales Sales Almeida, João Guilherme Alves Santos, Thiago Laitz, and Giovana Kerche Bonás. Ticket-bench: A kickoff for multilingual and regionalized agent evaluation. *arXiv preprint arXiv:2509.14477v1*, 2025. URL http://arxiv.org/abs/2509.14477v1.

[5] Frazier N. Baker, Daniel Adu-Ampratwum, Reza Averly, Botao Yu, Huan Sun, and Xia Ning. Larc: Towards human-level constrained retrosynthesis planning through an agentic framework. *arXiv preprint arXiv:2508.11860v1*, 2025. URL http://arxiv.org/abs/2508.11860v1.

[6] Sumanth Balaji, Piyush Mishra, Aashraya Sachdeva, and Suraj Agrawal. Beyond ivr: Benchmarking customer support llm agents for business-adherence. *arXiv preprint arXiv:2601.00596v1*, 2026. URL http://arxiv.org/abs/2601.00596v1.

[7] Nikolas Belle, Dakota Barnes, Alfonso Amayuelas, Ivan Bercovich, Xin Eric Wang, and William Wang. Agents of change: Self-evolving llm agents for strategic planning. *arXiv preprint arXiv:2506.04651v2*, 2025. URL http://arxiv.org/abs/2506.04651v2.

[8] Vamshi Krishna Bonagiri, Ponnurangam Kumaragurum, Khanh Nguyen, and Benjamin Plaut. Check yourself before you wreck yourself: Selectively quitting improves llm agent safety. *arXiv preprint arXiv:2510.16492v2*, 2025. URL http://arxiv.org/abs/2510.16492v2.

[9] Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. 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. *arXiv preprint arXiv:2511.16108v1*, 2025. URL http://arxiv.org/abs/2511.16108v1.

[10] Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Grounded test-time adaptation for llm agents. *arXiv preprint arXiv:2511.04847v3*, 2025. URL http://arxiv.org/abs/2511.04847v3.

[11] Binger Chen, Tacettin Emre Bök, Behnood Rasti, Volker Markl, and Begüm Demir. Remsa: An llm agent for foundation model selection in remote sensing. *arXiv preprint arXiv:2511.17442v1*, 2025. URL http://arxiv.org/abs/2511.17442v1.

[12] Chaoran Chen, Bingsheng Yao, Ruishi Zou, Wenyue Hua, Weimin Lyu, Yanfang Ye, Toby Jia-Jun Li, and Dakuo Wang. Towards a design guideline for rpa evaluation: A survey of large language model-based role-playing agents. *arXiv preprint arXiv:2502.13012v3*, 2025. URL http://arxiv.org/abs/2502.13012v3.

[13] Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your llm agents are temporally blind: The misalignment between tool use decisions and human time perception. *arXiv preprint arXiv:2510.23853v2*, 2025. URL http://arxiv.org/abs/2510.23853v2.

[14] Jae-Woo Choi, Hyungmin Kim, Hyobin Ong, Minsu Jang, Dohyung Kim, Jaehong Kim, and Youngwoo Yoon. Reactree: Hierarchical llm agent trees with control flow for long-horizon task planning. *arXiv preprint arXiv:2511.02424v1*, 2025. URL http://arxiv.org/abs/2511.02424v1.

[15] Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: A review of large language models as autonomous agents and

tool users. *arXiv preprint arXiv:2508.17281v2*, 2025. URL http://arxiv.org/abs/2508.17281v2.

[16] Zikun Cui, Tianyi Huang, Chia-En Chiang, and Cuiqianhe Du. Toward verifiable misinformation detection: A multi-tool llm agent framework. *arXiv preprint arXiv:2508.03092v1*, 2025. URL http://arxiv.org/abs/2508.03092v1.

[17] João Vitor de Carvalho Silva and Douglas G. Macharet. Can llm agents solve collaborative tasks? a study on urgency-aware planning and coordination. *arXiv preprint arXiv:2508.14635v1*, 2025. URL http://arxiv.org/abs/2508.14635v1.

[18] Jia-Kai Dong, I-Wei Huang, Chun-Tin Wu, and Yi-Tien Tsai. Etom: A five-level benchmark for evaluating tool orchestration within the mcp ecosystem. *arXiv preprint arXiv:2510.19423v2*, 2025. URL http://arxiv.org/abs/2510.19423v2.

[19] Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for llm agents. *arXiv preprint arXiv:2601.08012v1*, 2026. URL http://arxiv.org/abs/2601.08012v1.

[20] Changyu Du, Sebastian Esser, Stavros Nousias, and André Borrmann. Text2bim: Generating building models using a large language model-based multi-agent framework. *arXiv preprint arXiv:2408.08054v2*, 2024. URL http://arxiv.org/abs/2408.08054v2.

[21] Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. A survey on the optimization of large language model-based agents. *arXiv preprint arXiv:2503.12434v1*, 2025. URL http://arxiv.org/abs/2503.12434v1.

[22] Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. *arXiv preprint arXiv:2402.04253v1*, 2024. URL http://arxiv.org/abs/2402.04253v1.

[23] Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Tinyagent: Function calling at the edge. *arXiv preprint arXiv:2409.00608v3*, 2024. URL http://arxiv.org/abs/2409.00608v3.

[24] Junfeng Fang, Zijun Yao, Ruipeng Wang, Haokai Ma, Xiang Wang, and Tat-Seng Chua. We should identify and mitigate third-party safety risks in mcp-powered agent systems. *arXiv preprint arXiv:2506.13666v1*, 2025. URL http://arxiv.org/abs/2506.13666v1.

[25] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978v3*, 2025. URL http://arxiv.org/abs/2505.10978v3.

[26] Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros Maglaras, Abderrahmane Lakas, and Merouane Debbah. From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *arXiv preprint arXiv:2506.23260v2*, 2025. URL http://arxiv.org/abs/2506.23260v2.

[27] Xiaohan Fu, Shuheng Li, Zihan Wang, Yihao Liu, Rajesh K. Gupta, Taylor Berg-Kirkpatrick, and Earlence Fernandes. Imprompter: Tricking llm agents into improper tool use. *arXiv preprint arXiv:2410.14923v2*, 2024. URL http://arxiv.org/abs/2410.14923v2.

[28] Yuchuan Fu, Xiaohan Yuan, and Dongxia Wang. Ras-eval: A comprehensive benchmark for security evaluation of llm agents in real-world environments. *arXiv preprint arXiv:2506.15253v1*, 2025. URL http://arxiv.org/abs/2506.15253v1.

[29] Stefano Fumero, Kai Huang, Matteo Boffa, Danilo Giordano, Marco Mellia, Zied Ben Houidi, and Dario Rossi. Cybersleuth: Autonomous blue-team llm agent for web attack forensics. *arXiv preprint arXiv:2508.20643v1*, 2025. URL http://arxiv.org/abs/2508.20643v1.

[30] Silin Gao, Jane Dwivedi-Yu, Ping Yu, Xiaoqing Ellen Tan, Ramakanth Pasunuru, Olga Golovneva, Koustuv Sinha, Asli Celikyilmaz, Antoine Bosselut, and Tianlu Wang. Efficient tool use with chain-of-abstraction reasoning. *arXiv preprint arXiv:2401.17464v3*, 2024. URL http://arxiv.org/abs/2401.17464v3.

[31] Xuanqi Gao, Siyi Xie, Juan Zhai, Shiqing Ma, and Chao Shen. Mcp-radar: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *arXiv preprint arXiv:2505.16700v2*, 2025. URL http://arxiv.org/abs/2505.16700v2.

[32] Tarek Gasmi, Ramzi Guesmi, Ines Belhadj, and Jihene Bennaceur. Bridging ai and software security: A comparative vulnerability assessment of llm agent deployment paradigms. *arXiv preprint arXiv:2507.06323v1*, 2025. URL http://arxiv.org/abs/2507.06323v1.

[33] Eli Gendreau-Distler, Joshua Ho, Dongwon Kim, Luc Tomas Le Pottier, Haichen Wang, and Chengxi Yang. Automating high energy physics data analysis with llm-powered agents. *arXiv preprint arXiv:2512.07785v1*, 2025. URL http://arxiv.org/abs/2512.07785v1.

[34] Amur Ghose, Andrew B. Kahng, Sayak Kundu, and Zhiang Wang. Orfs-agent: Tool-using agents for chip design optimization. *arXiv preprint arXiv:2506.08332v2*, 2025. URL http://arxiv.org/abs/2506.08332v2.

[35] Jiale Guo, Suizhi Huang, Mei Li, Dong Huang, Xingsheng Chen, Regina Zhang, Zhijiang Guo, Han Yu, Siu-Ming Yiu, Pietro Lio, and Kwok-Yan Lam. A comprehensive survey on benchmarks and solutions in software engineering of llm-empowered agentic system. *arXiv preprint arXiv:2510.09721v3*, 2025. URL http://arxiv.org/abs/2510.09721v3.

[36] Tanmay Gupta, Luca Weihs, and Aniruddha Kembhavi. Codenav: Beyond tool-use to using real-world codebases with llm agents. *arXiv preprint arXiv:2406.12276v1*, 2024. URL http://arxiv.org/abs/2406.12276v1.

[37] Tsimur Hadeliya, Mohammad Ali Jauhar, Nidhi Sakpal, and Diogo Cruz. When refusals fail: Unstable safety mechanisms in long-context llm agents. *arXiv preprint arXiv:2512.02445v1*, 2025. URL http://arxiv.org/abs/2512.02445v1.

[38] Bingguang Hao, Zengzhuang Xu, Yuntao Wen, Xinyi Xu, Yang Liu, Tong Zhao, Maolin Wang, Long Chen, Dong Wang, Yicheng Chen, Cunyin Peng, Xiangyu Zhao, Chenyi Zhuang, and Ji Zhang. From failure to mastery: Generating hard samples for tool-use agents. *arXiv preprint arXiv:2601.01498v1*, 2026. URL http://arxiv.org/abs/2601.01498v1.

[39] Kostas Hatalis, Despina Christou, and Vyshnavi Kondapalli. Review of case-based reasoning for llm agents: Theoretical foundations, architectural components, and cognitive integration. *arXiv preprint arXiv:2504.06943v2*, 2025. URL http://arxiv.org/abs/2504.06943v2.

[40] Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S. Yu. The emerged security and privacy of llm agent: A survey with case studies. *arXiv preprint arXiv:2407.19354v2*, 2024. URL http://arxiv.org/abs/2407.19354v2.

[41] Yufei He, Ruoyu Li, Alex Chen, Yue Liu, Yulin Chen, Yuan Sui, Cheng Chen, Yi Zhu, Luca Luo, Frank Yang, and Bryan Hooi. Enabling self-improving agents to learn at test time with human-in-the-loop guidance. *arXiv preprint arXiv:2507.17131v2*, 2025. URL http://arxiv.org/abs/2507.17131v2.

[42] Joey Hong, Anca Dragan, and Sergey Levine. Planning without search: Refining frontier llms with offline goal-conditioned rl. *arXiv preprint arXiv:2505.18098v2*, 2025. URL http://arxiv.org/abs/2505.18098v2.

[43] Joey Hong, Kang Liu, Zhan Ling, Jiecao Chen, and Sergey Levine. Natural language actor-critic: Scalable off-policy learning in language space. *arXiv preprint arXiv:2512.04601v1*, 2025. URL http://arxiv.org/abs/2512.04601v1.

[44] Hanjiang Hu, Changliu Liu, Na Li, and Yebin Wang. Training task reasoning llm agents for multi-turn task planning via single-turn reinforcement learning. *arXiv preprint arXiv:2509.20616v2*, 2025. URL http://arxiv.org/abs/2509.20616v2.

[45] Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David A Ross, Cordelia Schmid, and Alireza Fathi. Avis: Autonomous visual information seeking with large language model agent. *arXiv preprint arXiv:2306.08129v3*, 2023. URL http://arxiv.org/abs/2306.08129v3.

[46] Jingyi Huang, Yuyi Yang, Mengmeng Ji, Charles Alba, Sheng Zhang, and Ruopeng An. Use of retrieval-augmented large language model agent for long-form covid-19 fact-checking. *arXiv preprint arXiv:2512.00007v1*, 2025. URL http://arxiv.org/abs/2512.00007v1.

[47] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716v1*, 2024. URL http://arxiv.org/abs/2402.02716v1.

[48] Changhyun Jeon, Jinhee Park, Jungwoo Choi, Keonwoo Kim, Jisu Kim, and Minji Hong. Slm-based agentic ai with p-c-g: Optimized for korean tool use. *arXiv preprint arXiv:2509.19369v1*, 2025. URL http://arxiv.org/abs/2509.19369v1.

[49] Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240v2*, 2025. URL http://arxiv.org/abs/2509.21240v2.

[50] Zhenlan Ji, Daoyuan Wu, Pingchuan Ma, Zongjie Li, and Shuai Wang. Testing and understanding erroneous planning in llm agents through synthesized user inputs. *arXiv preprint arXiv:2404.17833v1*, 2024. URL http://arxiv.org/abs/2404.17833v1.

[51] Zimo Ji, Xunguang Wang, Zongjie Li, Pingchuan Ma, Yudong Gao, Daoyuan Wu, Xincheng Yan, Tian Tian, and Shuai Wang. Taxonomy, evaluation and exploitation of ipi-centric llm agent defense frameworks. *arXiv preprint arXiv:2511.15203v1*, 2025. URL http://arxiv.org/abs/2511.15203v1.

[52] Jingyi Jia and Qinbin Li. Autotool: Efficient tool selection for large language model agents. *arXiv preprint arXiv:2511.14650v1*, 2025. URL http://arxiv.org/abs/2511.14650v1.

[53] Neil Kale, Chen Bo Calvin Zhang, Kevin Zhu, Ankit Aich, Paula Rodriguez, Scale Red Team, Christina Q. Knight, and Zifan Wang. Reliable weak-to-strong monitoring of llm agents. *arXiv preprint arXiv:2508.19461v1*, 2025. URL http://arxiv.org/abs/2508.19461v1.

[54] Doyoung Kim, Zhiwei Ren, Jie Hao, Zhongkai Sun, Lichao Wang, Xiyao Ma, Zack Ye, Xu Han, Jun Yin, Heng Ji, Wei Shen, Xing Fan, Benjamin Yao, and Chenlei Guo. Beyond perfect apis: A comprehensive evaluation of llm agents under real-world api complexity. *arXiv preprint arXiv:2601.00268v1*, 2026. URL http://arxiv.org/abs/2601.00268v1.

[55] Myung Ho Kim. Bridging symbolic control and neural reasoning in llm agents: The structured cognitive loop. *arXiv preprint arXiv:2511.17673v3*, 2025. URL http://arxiv.org/abs/2511.17673v3.

[56] Anis Koubaa and Khaled Gabr. Agentic uavs: Llm-driven autonomy with integrated tool-calling and cognitive reasoning. *arXiv preprint arXiv:2509.13352v2*, 2025. URL http://arxiv.org/abs/2509.13352v2.

[57] Mandar Kulkarni. Agent-s: Llm agentic workflow to automate standard operating procedures. *arXiv preprint arXiv:2503.15520v1*, 2025. URL http://arxiv.org/abs/2503.15520v1.

[58] Dae Cheol Kwon and Xinyu Zhang. Cp-agentnet: Autonomous and explainable communication protocol design using generative agents. *arXiv preprint arXiv:2503.17850v1*, 2025. URL http://arxiv.org/abs/2503.17850v1.

[59] Dawei Li, Yuguang Yao, Zhen Tan, Huan Liu, and Ruocheng Guo. Toolprmbench: Evaluating and advancing process reward models for tool-using agents. *arXiv preprint arXiv:2601.12294v1*, 2026. URL http://arxiv.org/abs/2601.12294v1.

[60] Jia Li, Xianjie Shi, Kechi Zhang, Ge Li, Zhi Jin, Lei Li, Huangzhao Zhang, Jia Li, Fang Liu, Yuwei Zhang, Zhengwei Tao, Yihong Dong, Yuqi Zhu, and Chongyang Tao. Graphcodeagent: Dual graph-guided llm agent for retrieval-augmented repo-level code generation. *arXiv preprint arXiv:2504.10046v2*, 2025. URL http://arxiv.org/abs/2504.10046v2.

[61] Weitang Li, Jiajun Ren, Lixue Cheng, and Cunxi Gong. Autonomous quantum simulation through large language model agents. *arXiv preprint arXiv:2601.10194v1*, 2026. URL http://arxiv.org/abs/2601.10194v1.

[62] Xinran Li, Chenjia Bai, Zijian Li, Jiakun Zheng, Ting Xiao, and Jun Zhang. Learn as individuals, evolve as a team: Multi-agent llms adaptation in embodied environments. *arXiv preprint arXiv:2506.07232v1*, 2025. URL http://arxiv.org/abs/2506.07232v1.

[63] Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use (including rag), planning, and feedback learning. *arXiv preprint arXiv:2406.05804v6*, 2024. URL http://arxiv.org/abs/2406.05804v6.

[64] Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, Fengli Xu, and Yong Li. Agentswift: Efficient llm agent design via value-guided hierarchical search. *arXiv preprint arXiv:2506.06017v2*, 2025. URL http://arxiv.org/abs/2506.06017v2.

[65] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459v2*, 2024. URL http://arxiv.org/abs/2401.05459v2.

[66] Yuran Li, Jama Hussein Mohamud, Chongren Sun, Di Wu, and Benoit Boulet. Leveraging llms as meta-judges: A multi-agent framework for evaluating llm judgments. *arXiv preprint arXiv:2504.17087v1*, 2025. URL http://arxiv.org/abs/2504.17087v1.

[67] Zhiwei Li, Yong Hu, and Wenqing Wang. Encouraging good processes without the need for good answers: Reinforcement learning for llm agent planning. *arXiv preprint arXiv:2508.19598v1*, 2025. URL http://arxiv.org/abs/2508.19598v1.

[68] Marianne Menglin Liu, Daniel Garcia, Fjona Parllaku, Vikas Upadhyay, Syed Fahad Allam Shah, and Dan Roth. Toolscope: Enhancing llm agent tool use through tool merging and context-aware filtering. *arXiv preprint arXiv:2510.20036v1*, 2025. URL http://arxiv.org/abs/2510.20036v1.

[69] Shuang Liu, Ruijia Zhang, Ruoyun Ma, Yujia Deng, Lanyi Zhu, Jiayu Li, Zelong Li, Zhibin Shen, and Mengnan Du. Llm agents in law: Taxonomy, applications, and challenges. *arXiv preprint arXiv:2601.06216v1*, 2026. URL http://arxiv.org/abs/2601.06216v1.

[70] Wenrui Liu, Zixiang Liu, Elsie Dai, Wenhan Yu, Lei Yu, Tong Yang, Jinjun Han, and Hong Gao. Mcpagentbench: A real-world task benchmark for evaluating llm agent mcp tool use. *arXiv preprint arXiv:2512.24565v3*, 2025. URL http://arxiv.org/abs/2512.24565v3.

[71] Yinqiu Liu, Ruichen Zhang, Haoxiang Luo, Yijing Lin, Geng Sun, Dusit Niyato, Hongyang Du, Zehui Xiong, Yonggang Wen, Abbas Jamalipour, Dong In Kim, and Ping Zhang. Secure multi-llm agentic ai and agentification for edge general intelligence by zero-trust: A survey. *arXiv preprint arXiv:2508.19870v1*, 2025. URL http://arxiv.org/abs/2508.19870v1.

[72] Siyuan Lu, Zechuan Wang, Hongxuan Zhang, Qintong Wu, Leilei Gan, Chenyi Zhuang, Jinjie Gu, and Tao Lin. Don't just fine-tune the agent, tune the environment. *arXiv preprint arXiv:2510.10197v1*, 2025. URL http://arxiv.org/abs/2510.10197v1.

[73] Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A. Burke. Memtool: Optimizing short-term memory management for dynamic tool calling in llm agent multi-turn conversations. *arXiv preprint arXiv:2507.21428v1*, 2025. URL http://arxiv.org/abs/2507.21428v1.

[74] 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. *arXiv preprint arXiv:2503.21460v1*, 2025. URL http://arxiv.org/abs/2503.21460v1.

[75] Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. Agrail: A lifelong agent guardrail with effective and adaptive safety detection. *arXiv preprint arXiv:2502.11448v2*, 2025. URL http://arxiv.org/abs/2502.11448v2.

[76] Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704v1*, 2025. URL http://arxiv.org/abs/2508.14704v1.

[77] Reza Yousefi Maragheh and Yashar Deldjoo. The future is agentic: Definitions, perspectives, and open challenges of multi-agent recommender systems. *arXiv preprint arXiv:2507.02097v2*, 2025. URL http://arxiv.org/abs/2507.02097v2.

[78] Rebecca Martin, Jay Patrikar, and Sebastian Scherer. Autoodd: Agentic audits via bayesian red teaming in black-box models. *arXiv preprint arXiv:2509.08638v1*, 2025. URL http://arxiv.org/abs/2509.08638v1.

[79] Panagiotis Michelakis, Yiannis Hadjiyiannis, and Dimitrios Stamoulis. Core: Full-path evaluation of llm agents beyond final state. *arXiv preprint arXiv:2509.20998v1*, 2025. URL http://arxiv.org/abs/2509.20998v1.

[80] Hyunji Min, Sangwon Jung, Junyoung Sung, Dosung Lee, Leekyeung Han, and Paul Hongsuck Seo. Goat: A training framework for goal-oriented agent with tools. *arXiv preprint arXiv:2510.12218v1*, 2025. URL http://arxiv.org/abs/2510.12218v1.

[81] Daisuke Miyamoto, Takuji Iimura, and Narushige Michishita. An llm agent-based framework for whaling countermeasures. *arXiv preprint arXiv:2601.14606v1*, 2026. URL http://arxiv.org/abs/2601.14606v1.

[82] Kanghua Mo, Li Hu, Yucheng Long, and Zhihao Li. Attractive metadata attack: Inducing llm agents to invoke malicious tools. *arXiv preprint arXiv:2508.02110v2*, 2025. URL http://arxiv.org/abs/2508.02110v2.

[83] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. *arXiv preprint arXiv:2507.21504v1*, 2025. URL http://arxiv.org/abs/2507.21504v1.

[84] Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. Toolsafe: Enhancing tool invocation safety of llm-based agents via proactive step-level guardrail and feedback. *arXiv preprint arXiv:2601.10156v1*, 2026. URL http://arxiv.org/abs/2601.10156v1.

[85] Nayantara Mudur, Hao Cui, Subhashini Venugopalan, Paul Raccuglia, Michael P. Brenner, and Peter Norgaard. Feabench: Evaluating language models on multiphysics reasoning ability. *arXiv preprint arXiv:2504.06260v1*, 2025. URL http://arxiv.org/abs/2504.06260v1.

[86] Katsuaki Nakano, Reza Fayyazi, Shanchieh Jay Yang, and Michael Zuzak. Guided reasoning in llm-driven penetration testing using structured attack trees. *arXiv preprint arXiv:2509.07939v2*, 2025. URL http://arxiv.org/abs/2509.07939v2.

[87] Subhrangshu Nandi, Arghya Datta, Nikhil Vichare, Indranil Bhattacharya, Huzefa Raja, Jing Xu, Shayan Ray, Giuseppe Carenini, Abhi Srivastava, Aaron Chan, Man Ho Woo, Amar Kandola, Brandon Theresa, and Francesco Carbone. Sop-bench: Complex industrial sops for evaluating llm agents. *arXiv preprint arXiv:2506.08119v1*, 2025. URL http://arxiv.org/abs/2506.08119v1.

[88] Khouloud Oueslati, Maxime Lamothe, and Foutse Khomh. Refagent: A multi-agent llm-based framework for automatic software refactoring. *arXiv preprint arXiv:2511.03153v1*, 2025. URL http://arxiv.org/abs/2511.03153v1.

[89] Ji-Lun Peng, Sijia Cheng, Egil Diau, Yung-Yu Shih, Po-Heng Chen, Yen-Ting Lin, and Yun-Nung Chen. A survey of useful llm evaluation. *arXiv preprint arXiv:2406.00936v1*, 2024. URL http://arxiv.org/abs/2406.00936v1.

[90] Yixiao Peng, Hao Hu, Feiyang Li, Xinye Cao, Yingchang Jiang, Jipeng Tang, Guoshun Nan, and Yuling Liu. Enhancing cloud network resilience via a robust llm-empowered multi-agent reinforcement learning framework. *arXiv preprint arXiv:2601.07122v1*, 2026. URL http://arxiv.org/abs/2601.07122v1.

[91] Aske Plaat, Max van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. Agentic large language models, a survey. *arXiv preprint arXiv:2503.23037v3*, 2025. URL http://arxiv.org/abs/2503.23037v3.

[92] Tian Qin, Felix Bai, Ting-Yao Hu, Raviteja Vemulapalli, Hema Swetha Koppula, Zhiyang Xu, Bowen Jin, Mert Cemri, Jiarui Lu, Zirui Wang, and Meng Cao. Compass: A multi-turn benchmark for tool-mediated planning & preference optimization. *arXiv preprint arXiv:2510.07043v1*, 2025. URL http://arxiv.org/abs/2510.07043v1.

[93] Santosh Kumar Radha, Yasamin Nouri Jelyani, Ara Ghukasyan, and Oktay Goktas. Iteration of thought: Leveraging inner dialogue for autonomous large language model reasoning. *arXiv preprint arXiv:2409.12618v2*, 2024. URL http://arxiv.org/abs/2409.12618v2.

[94] Mrinal Rawat, Ambuje Gupta, Rushil Goomer, Alessandro Di Bari, Neha Gupta, and Roberto Pieraccini. Pre-act: Multi-step planning and reasoning improves acting in llm agents. *arXiv preprint arXiv:2505.09970v2*, 2025. URL http://arxiv.org/abs/2505.09970v2.

[95] Matthew Russo and Tim Kraska. Deep research is the new analytics system: Towards building the runtime for ai-driven analytics. *arXiv preprint arXiv:2509.02751v1*, 2025. URL http://arxiv.org/abs/2509.02751v1.

[96] Abdelaziz Salama, Zeinab Nezami, Mohammed M. H. Qazzaz, Maryam Hafeez, and Syed Ali Raza Zaidi. Edge agentic ai framework for autonomous network optimisation in o-ran. *arXiv preprint arXiv:2507.21696v4*, 2025. URL http://arxiv.org/abs/2507.21696v4.

[97] Mohammad Hossein Samaei, Faryad Darabi Sahneh, Lee W. Cohnstaedt, and Caterina Scoglio. Epidemiqs: Prompt-to-paper llm agents for epidemic modeling and analysis. *arXiv preprint arXiv:2510.00024v1*, 2025. URL http://arxiv.org/abs/2510.00024v1.

[98] Anjana Sarkar and Soumyendu Sarkar. Survey of llm agent communication with mcp: A software design pattern centric review. *arXiv preprint arXiv:2506.05364v1*, 2025. URL http://arxiv.org/abs/2506.05364v1.

[99] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761v1*, 2023. URL http://arxiv.org/abs/2302.04761v1.

[100] Zeyang Sha, Hanling Tian, Zhuoer Xu, Shiwen Cui, Changhua Meng, and Weiqiang Wang. Agent safety alignment via reinforcement learning. *arXiv preprint arXiv:2507.08270v1*, 2025. URL http://arxiv.org/abs/2507.08270v1.

[101] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153v3*, 2024. URL http://arxiv.org/abs/2410.06153v3.

[102] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324v3*, 2024. URL http://arxiv.org/abs/2401.07324v3.

[103] Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Programmable privilege control for llm agents. *arXiv preprint arXiv:2504.11703v2*, 2025. URL http://arxiv.org/abs/2504.11703v2.

[104] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D. Wang. Ehragent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. *arXiv preprint arXiv:2401.07128v3*, 2024. URL http://arxiv.org/abs/2401.07128v3.

[105] Yuchen Shi, Yuzheng Cai, Siqi Cai, Zihan Xu, Lichao Chen, Yulei Qin, Zhijian Zhou, Xiang Fei, Chaofan Qiu, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Guocan Cai, Yong Mao, Yunsheng Wu, Ke Li, and Xing Sun. Youtu-agent: Scaling agent productivity with automated generation and hybrid policy optimization. *arXiv preprint arXiv:2512.24615v1*, 2025. URL http://arxiv.org/abs/2512.24615v1.

[106] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366v4*, 2023. URL http://arxiv.org/abs/2303.11366v4.

[107] Abdelrahman Soliman, Ahmed Refaey, Aiman Erbad, and Amr Mohamed. Intagent: Nwdaf-based intent llm agent towards advanced next generation networks. *arXiv preprint arXiv:2601.13114v1*, 2026. URL http://arxiv.org/abs/2601.13114v1.

[108] Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. *arXiv preprint arXiv:2601.05808v1*, 2026. URL http://arxiv.org/abs/2601.05808v1.

[109] Yueqi Song, Ketan Ramaneti, Zaid Sheikh, Ziru Chen, Boyu Gou, Tianbao Xie, Yiheng Xu, Danyang Zhang, Apurva Gandhi, Fan Yang, Joseph Liu, Tianyue Ou, Zhihao Yuan, Frank Xu, Shuyan Zhou, Xingyao Wang, Xiang Yue, Tao Yu, Huan Sun, Yu Su, and Graham Neubig. Agent data protocol: Unifying datasets for diverse, effective fine-tuning of llm agents. *arXiv preprint arXiv:2510.24702v1*, 2025. URL http://arxiv.org/abs/2510.24702v1.

[110] Hongyuan Tao, Ying Zhang, Zhenhao Tang, Hongen Peng, Xukun Zhu, Bingchang Liu, Yingguang Yang, Ziyin Zhang, Zhaogui Xu, Haipeng Zhang, Linchao Zhu, Rui Wang, Hang Yu, Jianguo Li, and Peng Di. Code graph model (cgm): A graph-integrated large language model for repository-level software engineering tasks. *arXiv preprint arXiv:2505.16901v4*, 2025. URL http://arxiv.org/abs/2505.16901v4.

[111] Vali Tawosi, Salwa Alamir, Xiaomo Liu, and Manuela Veloso. Meta-rag on large codebases using code summarization. *arXiv preprint arXiv:2508.02611v1*, 2025. URL http://arxiv.org/abs/2508.02611v1.

[112] Vali Tawosi, Keshav Ramani, Salwa Alamir, and Xiaomo Liu. Almas: an autonomous llm-based multi-agent software engineering framework. *arXiv preprint arXiv:2510.03463v2*, 2025. URL http://arxiv.org/abs/2510.03463v2.

[113] Elizaveta Tennant, Stephen Hailes, and Mirco Musolesi. Moral alignment for llm agents. *arXiv preprint arXiv:2410.01639v4*, 2024. URL http://arxiv.org/abs/2410.01639v4.

[114] Irene Testini, José Hernández-Orallo, and Lorenzo Pacchiardi. Measuring data science automation: A survey of evaluation tools for ai assistants and agents. *arXiv preprint arXiv:2506.08800v2*, 2025. URL http://arxiv.org/abs/2506.08800v2.

[115] Aleksandar Tomašević, Darja Cvetković, Sara Major, Slobodan Maletić, Miroslav Anđelković, Ana Vranić, Boris Stupovski, Dušan Vudragović, Aleksandar Bogojević, and Marija Mitrović Dankulov. Towards operational validation of llm-agent social simulations: A replicated study of a reddit-like technology forum. *arXiv preprint arXiv:2508.21740v2*, 2025. URL http://arxiv.org/abs/2508.21740v2.

[116] Arunkumar V, Gangadharan G. R., and Rajkumar Buyya. Agentic artificial intelligence (ai): Architectures, taxonomies, and evaluation of large language model agents. *arXiv preprint arXiv:2601.12560v1*, 2026. URL http://arxiv.org/abs/2601.12560v1.

[117] Minh-Hao Van, Prateek Verma, Chen Zhao, and Xintao Wu. A survey of ai for materials science: Foundation models, llm agents, datasets, and tools. *arXiv preprint arXiv:2506.20743v1*, 2025. URL http://arxiv.org/abs/2506.20743v1.

[118] Nikhil Verma. Active context compression: Autonomous memory management in llm agents. *arXiv preprint arXiv:2601.07190v1*, 2026. URL http://arxiv.org/abs/2601.07190v1.

[119] 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. *arXiv preprint arXiv:2305.16291v2*, 2023. URL http://arxiv.org/abs/2305.16291v2.

[120] He Wang, Alexander Hanbo Li, Yiqun Hu, Sheng Zhang, Hideo Kobayashi, Jiani Zhang, Henry Zhu, Chung-Wei Hang, and Patrick Ng. Dsmentor: Enhancing data science agents with curriculum learning and online knowledge accumulation. *arXiv preprint arXiv:2505.14163v1*, 2025. URL http://arxiv.org/abs/2505.14163v1.

[121] Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, Liang Lin, Zhihao Xu, Haolang Lu, Xinye Cao, Xinyun Zhou, Weifei Jin, Fanci Meng, Shicheng Xu, Junyuan Mao, Yu Wang, Hao Wu, Minghe Wang, Fan Zhang, Junfeng Fang, Wenjie Qu, Yue Liu, Chengwei Liu, Yifan Zhang, Qiankun Li, Chongye Guo, Yalan Qin, Zhaoxin Fan, Kai Wang, Yi Ding, Donghai Hong, Jiaming Ji, Yingxin Lai, Zitong Yu, Xinfeng Li, Yifan Jiang, Yanhui Li, Xinyu Deng, Junlin Wu, Dongxia Wang, Yihao Huang, Yufei Guo, Jen tse Huang, Qiufeng Wang, Xiaolong Jin, Wenxuan Wang, Dongrui Liu, Yanwei Yue, Wenke Huang, Guancheng Wan, Heng Chang, Tianlin Li, Yi Yu, Chenghao Li, Jiawei Li, Lei Bai, Jie Zhang, Qing Guo, Jingyi Wang, Tianlong Chen, Joey Tianyi Zhou, Xiaojun Jia, Weisong Sun, Cong Wu, Jing Chen, Xuming Hu, Yiming Li, Xiao Wang, Ningyu Zhang, Luu Anh Tuan, Guowen Xu, Jiaheng Zhang, Tianwei Zhang, Xingjun Ma, Jindong Gu, Liang Pang, Xiang Wang, Bo An, Jun Sun, Mohit Bansal, Shirui Pan, Lingjuan Lyu, Yuval Elovici, Bhavya Kailkhura, Yaodong Yang, Hongwei Li, Wenyuan Xu, Yizhou Sun, Wei Wang, Qing Li, Ke Tang, Yu-Gang Jiang, Felix Juefei-Xu, Hui Xiong, Xiaofeng Wang, Dacheng Tao, Philip S. Yu, Qingsong Wen, and Yang Liu. A comprehensive survey in llm(-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585v4*, 2025. URL http://arxiv.org/abs/2504.15585v4.

[122] Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, Jian Du, Yaxin Du, Xianghe Pang, Keduan Huang, Yanfeng Wang, Qiang Yan, and Siheng Chen. Mcp-flow: Facilitating llm agents to master real-world, diverse and scaling mcp tools. *arXiv preprint arXiv:2510.24284v2*, 2025. URL http://arxiv.org/abs/2510.24284v2.

[123] Chunlong Wu, Ye Luo, Zhibo Qu, and Min Wang. Meta-policy reflexion: Reusable reflective memory and rule admissibility for resource-efficient llm agent. *arXiv preprint arXiv:2509.03990v2*, 2025. URL http://arxiv.org/abs/2509.03990v2.

[124] Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. *arXiv preprint arXiv:2502.04644v2*, 2025. URL http://arxiv.org/abs/2502.04644v2.

[125] Panlong Wu, Kangshuo Li, Junbao Nan, and Fangxin Wang. Federated in-context llm agent learning. *arXiv preprint arXiv:2412.08054v1*, 2024. URL http://arxiv.org/abs/2412.08054v1.

[126] 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. *arXiv preprint arXiv:2510.16079v1*, 2025. URL http://arxiv.org/abs/2510.16079v1.

[127] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *arXiv preprint arXiv:2406.11200v3*, 2024. URL http://arxiv.org/abs/2406.11200v3.

[128] Yu Xia, Yiran Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. Sand: Boosting llm agents with self-taught action deliberation. *arXiv preprint arXiv:2507.07441v2*, 2025. URL http://arxiv.org/abs/2507.07441v2.

[129] Bufang Yang, Lilin Xu, Liekang Zeng, Yunqi Guo, Siyang Jiang, Wenrui Lu, Kaiwei Liu, Hancheng Xiang, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. Proagent: Harnessing on-demand sensory contexts for proactive llm agent systems. *arXiv preprint arXiv:2512.06721v1*, 2025. URL http://arxiv.org/abs/2512.06721v1.

[130] Chen Yang, Ran Le, Yun Xing, Zhenwei An, Zongchao Chen, Wayne Xin Zhao, Yang Song, and Tao Zhang. Toolmind technical report: A large-scale, reasoning-enhanced tool-use dataset. *arXiv preprint arXiv:2511.15718v2*, 2025. URL http://arxiv.org/abs/2511.15718v2.

[131] Fuyi Yang, Chenchen Ye, Mingyu Derek Ma, Yijia Xiao, Matthew Yang, and Wei Wang. Bioverge: A comprehensive benchmark and study of self-evaluating agents for biomedical hypothesis generation. *arXiv preprint arXiv:2511.08866v1*, 2025. URL http://arxiv.org/abs/2511.08866v1.

[132] Yingxuan Yang, Qiuying Peng, Jun Wang, Ying Wen, and Weinan Zhang. Llm-based multi-agent systems: Techniques and business perspectives. *arXiv preprint arXiv:2411.14033v2*, 2024. URL http://arxiv.org/abs/2411.14033v2.

[133] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, Weiwen Liu, Ying Wen, Yong Yu, and Weinan Zhang. A survey of ai agent protocols. *arXiv preprint arXiv:2504.16736v3*, 2025. URL http://arxiv.org/abs/2504.16736v3.

[134] Yiwei Yang, Aibo Hu, Yusheng Zheng, Brian Zhao, Xinqi Zhang, Dawei Xiang, Kexin Chu, Wei Zhang, and Andi Quinn. Mvvm: Deploy your ai agents-securely, efficiently, everywhere. *arXiv preprint arXiv:2410.15894v2*, 2024. URL http://arxiv.org/abs/2410.15894v2.

[135] Taro Yano, Yoichi Ishibashi, and Masafumi Oyamada. Lamdagent: An autonomous framework for post-training pipeline optimization via llm agents. *arXiv preprint arXiv:2505.21963v1*, 2025. URL http://arxiv.org/abs/2505.21963v1.

[136] Huanjin Yao, Ruifei Zhang, Jiaxing Huang, Jingyi Zhang, Yibo Wang, Bo Fang, Ruolin Zhu, Yongcheng Jing, Shunyu Liu, Guanbin Li, and Dacheng Tao. A survey on agentic multimodal large language models. *arXiv preprint arXiv:2510.10991v1*, 2025. URL http://arxiv.org/abs/2510.10991v1.

[137] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629v3*, 2022. URL http://arxiv.org/abs/2210.03629v3.

[138] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601v2*, 2023. URL http://arxiv.org/abs/2305.10601v2.

[139] Yu Yao, Salil Bhatnagar, Markus Mazzola, Vasileios Belagiannis, Igor Gilitschenski, Luigi Palmieri, Simon Razniewski, and Marcel Hallgarten. Agents-llm: Augmentative generation of challenging traffic scenarios with an agentic llm framework. *arXiv preprint arXiv:2507.13729v1*, 2025. URL http://arxiv.org/abs/2507.13729v1.

[140] Ye Ye. Task memory engine (tme): Enhancing state awareness for multi-step llm agent tasks. *arXiv preprint arXiv:2504.08525v4*, 2025. URL http://arxiv.org/abs/2504.08525v4.

[141] Ye Ye. Task memory engine: Spatial memory for robust multi-step llm agents. *arXiv preprint arXiv:2505.19436v1*, 2025. URL http://arxiv.org/abs/2505.19436v1.

[142] Yauwai Yim, Chunkit Chan, Tianyu Shi, Zheye Deng, Wei Fan, Tianshi Zheng, and Yangqiu Song. Evaluating and enhancing llms agent based on theory of mind in guandan: A multi-player cooperative game under imperfect information. *arXiv preprint arXiv:2408.02559v1*, 2024. URL http://arxiv.org/abs/2408.02559v1.

[143] Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long T. Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. *arXiv preprint arXiv:2503.07826v1*, 2025. URL http://arxiv.org/abs/2503.07826v1.

[144] Guoli Yin, Haoping Bai, Shuang Ma, Feng Nan, Yanchao Sun, Zhaoyang Xu, Shen Ma, Jiarui Lu, Xiang Kong, Aonan Zhang, Dian Ang Yap, Yizhe zhang, Karsten Ahnert, Vik Kamath, Mathias Berglund, Dominic Walsh, Tobias Gindele, Juergen Wiest, Zhengfeng Lai, Xiaoming Wang, Jiulong Shan, Meng Cao, Ruoming Pang, and Zirui Wang. Mmau: A holistic benchmark of agent capabilities across diverse domains. *arXiv preprint arXiv:2407.18961v3*, 2024. URL http://arxiv.org/abs/2407.18961v3.

[145] Chenglin Yu, Yang Yu, Songmiao Wang, Yucheng Wang, Yifan Yang, Jinjia Li, Ming Li, and Hongxia Yang. Infiagent: Self-evolving pyramid agent framework for infinite scenarios. *arXiv preprint arXiv:2509.22502v2*, 2025. URL http://arxiv.org/abs/2509.22502v2.

[146] 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. *arXiv preprint arXiv:2601.01885v1*, 2026. URL http://arxiv.org/abs/2601.01885v1.

[147] Yuchen Zeng, Shuibai Zhang, Wonjun Kang, Shutong Wu, Lynnix Zou, Ying Fan, Heeju Kim, Ziqian Lin, Jungtaek Kim, Hyung Il Koo, Dimitris Papailiopoulos, and Kangwook Lee. Rejump: A tree-jump representation for analyzing and improving llm reasoning. *arXiv preprint arXiv:2512.00831v2*, 2025. URL http://arxiv.org/abs/2512.00831v2.

[148] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279v12*, 2024. URL http://arxiv.org/abs/2411.18279v12.

[149] Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. Datascibench: An llm agent benchmark for

data science. *arXiv preprint arXiv:2502.13897v1*, 2025. URL http://arxiv.org/abs/2502.13897v1.

[150] Dongsen Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li, and Wenjun Xu. Mcp security bench (msb): Benchmarking attacks against model context protocol in llm agents. *arXiv preprint arXiv:2510.15994v1*, 2025. URL http://arxiv.org/abs/2510.15994v1.

[151] Guibin Zhang, Haiyang Yu, Kaiming Yang, Bingli Wu, Fei Huang, Yongbin Li, and Shuicheng Yan. Evoroute: Experience-driven self-routing llm agent systems. *arXiv preprint arXiv:2601.02695v1*, 2026. URL http://arxiv.org/abs/2601.02695v1.

[152] Shaokun Zhang, Jieyu Zhang, Dujian Ding, Mirian Hipolito Garcia, Ankur Mallick, Daniel Madrigal, Menglin Xia, Victor Rühle, Qingyun Wu, and Chi Wang. Ecoact: Economic agent determines when to register what action. *arXiv preprint arXiv:2411.01643v1*, 2024. URL http://arxiv.org/abs/2411.01643v1.

[153] Xiaowen Zhang, Zhenyu Bi, Patrick Lachance, Xuan Wang, Tiziana Di Matteo, and Rupert A. C. Croft. Bridging literature and the universe via a multi-agent large language model system. *arXiv preprint arXiv:2507.08958v2*, 2025. URL http://arxiv.org/abs/2507.08958v2.

[154] Xin Zhang, Lissette Iturburu, Juan Nicolas Villamizar, Xiaoyu Liu, Manuel Salmeron, Shirley J. Dyke, and Julio Ramirez. Large language model agent for structural drawing generation using react prompt engineering and retrieval augmented generation. *arXiv preprint arXiv:2507.19771v1*, 2025. URL http://arxiv.org/abs/2507.19771v1.

[155] Xinyu Zhang, Yixin Wu, Boyang Zhang, Chenhao Lin, Chao Shen, Michael Backes, and Yang Zhang. Geo-detective: Unveiling location privacy risks in images with llm agents. *arXiv preprint arXiv:2511.22441v1*, 2025. URL http://arxiv.org/abs/2511.22441v1.

[156] Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. *arXiv preprint arXiv:2404.05427v3*, 2024. URL http://arxiv.org/abs/2404.05427v3.

[157] Zheng Zhang, Nuoqian Xiao, Qi Chai, Deheng Ye, and Hao Wang. Multimind: Enhancing werewolf agents with multimodal reasoning and theory of mind. *arXiv preprint arXiv:2504.18039v4*, 2025. URL http://arxiv.org/abs/2504.18039v4.

[158] Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470v2*, 2024. URL http://arxiv.org/abs/2412.14470v2.

[159] Lina Zhao, Jiaxing Bai, Zihao Bian, Qingyue Chen, Yafang Li, Guangbo Li, Min He, Huaiyuan Yao, and Zongjiu Zhang. Autonomous multi-modal llm agents for treatment planning in focused ultrasound ablation surgery. *arXiv preprint arXiv:2505.21418v2*, 2025. URL http://arxiv.org/abs/2505.21418v2.

[160] Yuheng Zhao, Junjie Wang, Linbin Xiang, Xiaowen Zhang, Zifei Guo, Cagatay Turkay, Yu Zhang, and Siming Chen. Lightva: Lightweight visual analytics with llm agent-based task planning and execution. *arXiv preprint arXiv:2411.05651v2*, 2024. URL http://arxiv.org/abs/2411.05651v2.

[161] Tianshi Zheng, Kelvin Kiu-Wai Tam, Newt Hue-Nam K. Nguyen, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Y. Wong, and Simon See. Newtonbench: Benchmarking generalizable scientific law discovery in llm agents. *arXiv preprint arXiv:2510.07172v2*, 2025. URL http://arxiv.org/abs/2510.07172v2.

[162] Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. Rtbas: Defending llm agents against prompt injection and privacy leakage. *arXiv preprint arXiv:2502.08966v2*, 2025. URL http://arxiv.org/abs/2502.08966v2.

[163] Kaiwen Zhou, Ahmed Elgohary, A S M Iftekhar, and Amin Saied. Siraj: Diverse and efficient red-teaming for llm agents via distilled structured reasoning. *arXiv preprint arXiv:2510.26037v1*, 2025. URL http://arxiv.org/abs/2510.26037v1.

[164] Kaiyu Zhou, Yongsen Zheng, Yicheng He, Meng Xue, Xueluan Gong, Yuji Wang, and Kwok-Yan Lam. Beyond max tokens: Stealthy resource amplification via tool calling chains in llm agents. *arXiv preprint arXiv:2601.10955v1*, 2026. URL http://arxiv.org/abs/2601.10955v1.

[165] Xingfu Zhou and Pengfei Wang. Reasoning-style poisoning of llm agents via stealthy style transfer: Process-level attacks and runtime monitoring in rsv space. *arXiv preprint arXiv:2512.14448v1*, 2025. URL http://arxiv.org/abs/2512.14448v1.

[166] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446v1*, 2024. URL http://arxiv.org/abs/2402.19446v1.

[167] Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *arXiv preprint arXiv:2506.01716v1*, 2025. URL http://arxiv.org/abs/2506.01716v1.

[168] Hanlin Zhu, Tianyu Guo, Song Mei, Stuart Russell, Nikhil Ghosh, Alberto Bietti, and Jiantao Jiao. Gsm-agent: Understanding agentic reasoning using controllable environments. *arXiv preprint arXiv:2509.21998v2*, 2025. URL http://arxiv.org/abs/2509.21998v2.

[169] Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370v1*, 2025. URL http://arxiv.org/abs/2509.25370v1.

[170] Yuchen Zhuang, Jingfeng Yang, Haoming Jiang, Xin Liu, Kewei Cheng, Sanket Lokegaonkar, Yifan Gao, Qing Ping, Tianyi Liu, Binxuan Huang, Zheng Li, Zhengyang Wang, Pei Chen, Ruijie Wang, Rongzhi Zhang, Nasser Zalmout, Priyanka Nigam, Bing Yin, and Chao Zhang. Hephaestus: Improving fundamental agent capabilities of large language models through continual pre-training. *arXiv preprint arXiv:2502.06589v1*, 2025. URL http://arxiv.org/abs/2502.06589v1.