agent survey latex

January 25, 2026

Contents

Abstract

Large language models increasingly act as closed-loop agents that iteratively perceive, decide, and act through tools and environments, rather than producing single-turn answers [127, 78, 146]. Yet reported performance often conflates model capability with interface contracts, tool access assumptions, and evaluation protocols, making cross-paper comparisons brittle [37, 92, 137]. This paper organizes the design space around foundations and interfaces, core components for planning and memory, adaptation and multi-agent coordination, and evaluation and risks, emphasizing protocol-aware contrasts over per-paper summaries [28, 62]. Across these axes, we highlight recurring failure modes in tool orchestration and long-horizon loops, and we summarize mitigation patterns for reliability and security [110, 136, 32]. Appendix tables compress representative approaches and protocol anchors to support fast comparison [77, 80, 107].

1 Introduction

Tool-augmented large language models are increasingly deployed as agents that must make sequences of decisions under partial observability, cost constraints, and unreliable tool execution [127, 78, 146, 115, 20]. In such settings, errors compound: a single mistaken tool call can derail downstream reasoning, while aggressive exploration can create new safety and security surfaces [110, 136, 32].

A useful starting point is to treat an LLM agent as a closed-loop system: it maintains an internal state (explicit or implicit), selects actions, observes outcomes, and updates its state in response [14, 39, 48, 150, 26]. This perspective separates agentic behavior from single-turn tool use by focusing on the loop structure and the assumptions that make the loop stable (e.g., observability, resetability, and action semantics) [58, 38].

Interface contracts largely determine what an agent can do and what a result means: schemas and orchestration policies constrain the action space, while permissions and sandboxing determine which actions are feasible at deployment time [138, 72, 46, 22, 66]. Seemingly small interface choices can change failure modes (silent tool misfires, name collisions, or unintended capability escalation), which in turn changes which evaluation claims are interpretable [53, 154].

Within the loop, core components such as planning, reasoning, and memory retrieval mediate long-horizon behavior, but they interact with protocol constraints in non-obvious ways [36, 19, 29, 97, 147]. Planning modules that look strong under generous tool access may collapse under stricter budgets or noisier tools, whereas memory policies can trade short-term success for robustness against prompt and data contamination [110, 93].

Evaluation remains a central bottleneck. Agent benchmarks vary widely in task families, success metrics, tool access assumptions, and reporting practices, and such heterogeneity can make head-to-head comparisons fragile unless protocols are normalized [37, 92, 137, 38, 44]. As a result, surveys that primarily list systems without making protocol context explicit often understate uncertainty and overstate generality [139, 25].

Risks are inseparable from design choices. Tool interfaces expand the attack surface through prompt injection, data exfiltration, and privilege misuse, and defenses that work for one interface contract may not transfer to another [136, 32, 110, 113, 140]. Governance and safety therefore depend on making assumptions explicit (threat models, tool boundaries, monitoring) rather than treating agent safety as a monolithic property [154, 53].

We compiled this survey by retrieving and deduplicating 1800 arXiv records for LLM-agent and tool-use queries (no strict time-window filter), then selecting a 300-paper core set for synthesis. Evidence is primarily abstract-level: claims are written conservatively when protocol details are missing, and quantitative statements are only used when minimal task, metric, and constraint context can be traced to the cited work [37, 28].

The remainder of the paper connects systems to the assumptions that drive their reported behavior. We first cover foundations and interfaces (agent loops, tool protocols), then planning

and memory components, then adaptation and coordination mechanisms, and finally evaluation and risk surfaces. Appendix tables summarize representative approaches and protocol anchors to support quick cross-section comparison [77, 80, 107, 112, 1].

2 Related Work

Several recent surveys and reviews chart the rapid growth of LLM-based agents, tool use, and evaluation practices, providing useful snapshots of emerging capabilities and application areas [37, 28, 62, 33, 141]. Their organizing principles vary (task categories, component taxonomies, or application verticals), and many emphasize breadth over protocol-aligned comparisons [139].

Work on tool interfaces and orchestration studies how action spaces are exposed to language models (e.g., function calling and protocolized tool APIs) and how agents select, sequence, and verify tool calls [138, 46, 59, 22, 72, 77]. These lines highlight that interface contracts (schema design, grounding, permissions) are often the dominant determinant of reliability, yet are inconsistently reported across benchmarks [31, 25].

System papers that focus on the agent loop often frame agents as iterative decision-makers that interleave reasoning traces with actions and observations, producing behavior that cannot be evaluated with static QA-style metrics alone [127, 78, 146, 117, 149]. Architectural choices such as where memory and planning live, how tool failures are handled, and whether intermediate states are exposed shape both performance and failure modes [14, 94].

Planning and reasoning loops span approaches from training specialized planners to steering inference-time deliberation, often trading off compute cost, controllability, and robustness under environment noise [36, 56, 105, 19, 147, 109]. A recurring challenge is that planning gains can be protocol-dependent: tool budgets, verification access, and horizon length can flip which method appears superior [38].

Memory and retrieval-augmented agents extend the loop with mechanisms for state persistence and evidence access, including retrieval policies and representations that mediate what context is available at decision time [29, 97, 121, 93, 110, 51]. This body of work underscores that more context is not universally better: contamination, prompt injection, and stale memories can degrade reliability unless protocols define what information is trusted [110].

Self-improvement and adaptation mechanisms modify policies, prompts, or tools based on feedback, ranging from offline prompt and context optimization to online reflection and revision loops [140, 90, 125, 115, 10, 139]. These approaches raise stability questions (reward hacking, overfitting to benchmarks) that are easy to miss when evaluation protocols are underspecified [38, 44].

Multi-agent systems introduce additional degrees of freedom such as role protocols, communication channels, and aggregation rules. These mechanisms can improve coverage and verification but also create new failure modes (collusion, feedback loops, cost blow-ups) [47, 1, 80, 43, 26, 112]. Many evaluations in this area rely on rubric-based or task-specific protocols, making it important to track interaction budgets and aggregation assumptions when comparing results [80].

Benchmark suites and evaluation protocols attempt to make agent behavior comparable across environments, but they vary in what constitutes success, how tool access is granted, and how long-horizon credit is assigned [37, 92, 137, 38, 44, 135]. Recent work also emphasizes reproducibility issues such as leakage, hidden environment changes, and dependence on proprietary tool backends, which can invalidate cross-paper rankings if not disclosed [139, 25].

Safety and security work argues that deploying tool-using agents requires explicit threat models and interface-aware mitigations, since attacks and defenses depend on the system boundary between model, tools, and external resources [136, 32, 53, 110, 113, 154]. This line connects evaluation to governance: metrics that ignore misuse and exfiltration can overstate capability while understating risk [136].

Across this literature, a persistent obstacle is that comparisons are frequently made without making protocol context and interface assumptions explicit. The organizing choice in this paper is to treat interfaces and protocols as first-class comparison objects, so that contrasts are anchored in task, metric, budget, and tool-access statements rather than in narrative summaries of individual papers [37, 28, 77, 136, 137].

3 Foundations & Interfaces

Interface choices largely determine what an agent can reliably do and what its measured performance means. Viewing agents as closed-loop systems makes it natural to treat action spaces, observation channels, and tool protocols as first-class design variables rather than peripheral implementation details [127, 138].

This chapter therefore separates (i) the agent loop and action space abstraction from (ii) the concrete tool interface and orchestration layer. The first subsection focuses on how loop assumptions shape failure recovery and evaluation comparability, while the second focuses on how schema design, routing, and protocolization affect reliability and safety under repeated interactions [77, 72].

3.1 Agent loop and action spaces

A central tension in agent loops is that richer action spaces promise broader capability, but they also make behavior harder to constrain, reproduce, and compare under realistic budgets. When evaluation protocols differ (task families, metrics, interaction budgets, tool access), apparent improvements can reflect protocol choices rather than methodological advantages, so comparisons only become meaningful after protocol context is made explicit [37, 53, 84, 65]. Accordingly, action spaces should be discussed together with the evaluation setup they imply, and protocol alignment should be treated as a prerequisite for cross-paper synthesis [137].

An agent loop can be modeled as a repeated state-decide-act-observe cycle, but the semantics of ”act” vary widely in tool-using settings. Actions may be direct environment steps, whereas tool-using agents often execute structured tool calls with schemas or mixed actions that combine invocation with natural-language control; each choice changes what is observable and what failure recovery is possible [127, 154, 48, 55, 79]. In practice, the action interface also determines what can be audited (e.g., whether intermediate tool arguments are visible) and which failures are recoverable versus silent [53, 64].

Action spaces are inseparable from the observation model. If observations are lossy or delayed, agents may rely on implicit state and brittle heuristics; if observations are rich, agents can support verification and correction loops, but at higher compute and latency cost [30, 137, 68]. This trade-off is often hidden in benchmark reporting, where small differences in logging, permissions, or tool visibility can change both success rates and the distribution of failure modes [93, 154, 131].

The benchmark landscape itself reveals why protocol context matters. A broad survey of agent evaluation spans 190+ benchmark datasets and documents a shift from static exams toward process- and discovery-oriented assessments, where interaction protocols and budgets become first-order variables [37, 114]. Under such diversity, ”agent loop quality” cannot be treated as a single scalar: claims must be interpreted relative to task family, metric definition, and the constraints imposed by the environment and tool interface [53, 71].

Recent benchmark-driven agent frameworks illustrate the consequence of wide, heterogeneous evaluation. AgentSwift evaluates across seven benchmarks spanning embodied, math, web, tool, and game domains, reporting an average improvement of 8.34% over baselines in that multi-domain setting [63]. AgentSquare reports gains across six benchmarks in similarly

diverse scenarios, but the comparability of these numbers depends on whether the underlying protocols (tool access, budgets, environment versions) are aligned across methods [92, 37].

Targeted dataset design provides a complementary angle: in contrast to broad benchmark suites, some work introduces bespoke datasets to stress specific loop properties. For example, MuaLLM introduces custom datasets aimed at retrieval and citation behavior and multi-step reasoning, which can surface action-space assumptions that are easy to miss in broader suites [1]. Such datasets can improve diagnostic power, but they also increase the risk of overfitting unless they are connected back to shared protocol anchors used elsewhere [37, 53].

Security and robustness further complicate the notion of an action space. Memory and tool interactions can act as long-lived state, which creates attack channels where adversarial content influences future decisions; defenses therefore become part of the loop design rather than an external add-on [110]. A-MemGuard reports large reductions in attack success rates with minimal utility cost, illustrating that loop safety can be evaluated as a first-class objective, but only when threat models and interaction assumptions are made explicit [110, 154].

Compute, latency, and monetary cost impose another unavoidable constraint. The agent system trilemma formalizes the tension among state-of-the-art performance, low cost, and fast completion, suggesting that "better loops" are often better only under a particular budget regime [137]. This motivates protocol-aware reporting: without budgets and time constraints, comparisons across action-space designs can be misleading even when they cite the same task label [53, 37].

Across these threads, a consistent pattern is that evaluation and action design co-determine each other. Loop abstractions that perform well on interactive benchmarks can fail under stricter tool permissions, while security-aware interfaces can change both capability and measured utility [127, 110, 137]. For synthesis, the conservative move is to compare methods only within shared protocol envelopes, and otherwise describe differences as protocol-driven shifts in what the loop can reliably guarantee [37, 53].

One limitation is that many papers still underspecify protocol details (budgets, tool access, visibility), which forces any survey-level comparison to weaken claims rather than guess missing context [37, 53]. Another is that action spaces are implemented through concrete tool interfaces and orchestration policies; without examining those interfaces, loop-level claims remain underspecified, which motivates the next subsection's focus on tool contracts and orchestration patterns [154, 53].

While loop design determines what actions are possible, tool interfaces determine how those actions are grounded in executable APIs and orchestration policies.

3.2 Tool interfaces and orchestration

For system builders, the crux of tool interface design is choosing how much expressivity to expose while keeping behavior controllable and verifiable. Richer interfaces can expand the action space, but they also increase the burden of protocol specification (schemas, permissions, observability) and widen the gap between what is evaluated and what is deployable [138, 72, 104]. Interface contracts largely determine what can be compared and what tool-use claims mean; meaningful comparisons therefore require explicit tool-access assumptions and reporting conventions rather than generic "tool use" narratives [22, 53, 148].

Tool interfaces range from free-form natural-language tool descriptions to structured function signatures and protocolized multi-tool stacks. As interfaces become more structured, they enable stronger verification and orchestration policies (routing, retries, argument validation), but they also introduce new brittleness: schema mismatches and implicit permissions can silently change the agent's effective action space [66, 138, 130, 153]. This makes interface design inseparable from evaluation: the same model can look strong or weak depending on which tool errors are detectable and recoverable under the benchmark protocol [149, 53].

Concrete protocolized systems illustrate how evaluation must name the interface details. MemTool evaluates multiple tool modes across 13+ LLMs on the ScaleMCP benchmark, using long interaction traces (100+ consecutive user interactions) and metrics such as tool removal ratios and task completion accuracy [77]. Such settings are useful precisely because they stress long-horizon orchestration behavior and reveal failure modes that short-horizon tasks can hide [77, 31].

Complementing system-driven evaluations, benchmark and dataset audits provide a map of what is actually being measured. An analysis of dozens of publicly available datasets highlights wide variation in task families and assessment protocols for LLM agents, suggesting that many comparisons are dominated by protocol mismatch rather than method differences [22]. This motivates a protocol-first reading: before comparing orchestration mechanisms, one should ask whether benchmarks share the same tool access assumptions and scoring rules [22, 53].

A useful contrast is between protocol-first tool stacks and ad hoc tool calling. Protocolized interfaces can make tool availability and permissions explicit (which supports reproducibility), whereas ad hoc interfaces often leave such details implicit and encourage "best-effort" behavior that is hard to audit [72, 138, 106]. In practice, the right comparison is therefore not "tool use vs no tool use" but "which interface constraints are enforced, and what failure recovery is allowed under the protocol" [77, 22].

Retrieval-augmented orchestration provides another contrast class. In contrast to ad hoc tool calling, systems such as AvaTaR exploit retrieval and tool selection under a fixed interface contract, reporting gains on retrieval-style settings (e.g., Hit@1 improvements on retrieval datasets) within that evaluation setup [117]. These results are only interpretable when the retrieval protocol is pinned down (candidate pool, metric definition, tool access), which again reinforces the need for protocol-aware reporting in the interface layer [117, 53].

Classic reasoning-action interleaving remains a useful baseline for tool interface discussions because it exposes how interface constraints change loop behavior. On interactive decision-making benchmarks (ALFWorld and WebShop), ReAct reports large absolute success gains over imitation and reinforcement baselines under a few-shot prompting regime [127]. Such gains can disappear when tool access policies, step budgets, or evaluation environments shift, which is why later benchmarks and protocol papers emphasize explicit reporting of those constraints [22, 146].

Security and cost failures can originate directly from the agent-tool communication loop. The interface is a critical attack surface, enabling multi-turn attacks that can inflate token usage, monetary costs, and energy consumption even when a task appears to be completed correctly [146]. Reported evaluations show trajectories exceeding 60,000 tokens and cost inflation up to 658x on common tool benchmarks, underscoring that orchestration must be assessed not only for success but also for budget-aware robustness [146, 31].

Across these studies, the most stable synthesis is to compare orchestration choices within shared protocol envelopes and to treat protocol differences as the primary confounder otherwise. Protocol papers explicitly recommend anchoring claims with task type, metric, and a concrete constraint (budget, tool access, horizon, or threat model), and avoiding underspecified baseline naming when details are missing [146, 22, 53]. This is also where benchmark surveys and suites become valuable: they provide a common vocabulary for what an interface actually constrains [37, 92].

One limitation is that interface descriptions often omit operational constraints (permissioning, sandbox boundaries, failure recovery), which makes post hoc comparison unreliable even when citations are plentiful [22, 53]. A second is that many benchmarks still under-measure security and cost failures, so orchestration systems can over-optimize for success metrics while remaining fragile to interface-layer attacks, motivating later sections on evaluation protocols and risks [146, 59].

4 Core Components (Planning + Memory)

Long-horizon behavior depends on how an agent forms plans and how it maintains usable state over time. Planning and reasoning modules can change action selection, but their benefits are often protocol-dependent: tool budgets, verification access, and environment noise can flip apparent rankings [36, 19].

Memory mechanisms provide persistence and evidence access, but they also introduce new failure modes when stale or adversarial information enters the loop. The two subsections in this chapter examine these components jointly: planning and reasoning loops as a control policy, and memory and retrieval as a state substrate whose trust assumptions must be explicit for comparisons to remain meaningful [29, 110].

4.1 Planning and reasoning loops

Results in planning-heavy agent loops are only comparable when the evaluation protocol is explicit: task family, metric, interaction horizon, and budget constraints jointly determine which reasoning strategy is viable. Under loose constraints, long chains of thought or multi-agent deliberation can look strong, whereas under strict budgets, simpler policies or direct tool-based solutions can dominate [38, 44, 145, 35]. The practical implication is that synthesis should treat protocol context as a first-class variable and compare planning methods within shared constraints rather than across incompatible setups [38, 44].

Planning and reasoning loops cover a spectrum of mechanisms that decide what to do next and how to justify it. Some approaches emphasize training or fine-tuning a planner, others steer inference-time deliberation, and still others treat planning as a program synthesis problem where tool use or code becomes the dominant reasoning substrate [36, 19, 96, 41]. These choices affect not only success rates but also failure recovery: a loop that can externalize intermediate plans can support verification and correction, but it also increases latency and cost [147, 10, 76].

A useful contrast is between training-based planners and steering-based planners. Training can bake in long-horizon behaviors and reduce prompt brittleness; in contrast, steering focuses on controlling inference-time trajectories (e.g., preventing overthinking or underthinking) without requiring retraining [36, 19]. In practice, the evaluation protocol decides which contrast matters: when tasks reward concise, tool-backed solutions, steering toward direct coding can outperform elaborate reasoning chains even at 100% success on some benchmarks [19, 109].

Concrete evidence of protocol sensitivity appears in planning benchmarks. A 1.5B-parameter model trained with single-turn GRPO reports strong long-horizon planning performance (70% success on a complex task planning benchmark), exceeding larger baselines up to 14B parameters under that benchmark's setup [36, 82]. Such results are informative only when the benchmark defines the horizon, action interface, and success metric clearly; otherwise, the same method can degrade when tool access, budgets, or environment noise change [38, 44].

Benchmark construction further shapes what "planning" means. USTBench evaluates spatiotemporal reasoning as an urban-agent capability across multiple dimensions (understanding, forecasting, planning, reflection with feedback), illustrating that planning may require structured state rather than free-form reasoning text [56]. This supports a protocol-first comparison: differences in observation structure and action affordances can be more predictive of planning success than differences in the reasoning module alone [56, 38, 4].

Planning often interacts with tool use and code in ways that undermine naive comparisons. Some tasks that appear to demand sophisticated reasoning can be solved by direct coding, which shifts the relevant question from "reasoning quality" to "whether the loop can select and verify the right tool or program under constraints" [19, 105, 54]. Accordingly, evaluation should report not only success but also budgets (token budget, cost, and latency), tool access, and verification channels, since these determine whether planning is doing substantive work or merely producing verbose traces [38, 44].

Memory and retrieval can make planning look better or worse depending on how evidence is supplied to the loop. Systems that treat retrieval as an explicit action (retrieve, reflect, or answer routing) effectively expand the planning action space, which can improve answer quality but also introduce new failure modes when retrieval is noisy or adversarial [29, 127]. For synthesis, this means planning results should be read together with the memory protocol: what information is accessible, when it is retrieved, and how it is validated [29, 38].

Failure modes in reasoning loops can also be adversarial rather than purely capability-driven. Process-oriented attacks that target reasoning style (e.g., inducing "analysis paralysis" or "cognitive haste") highlight that even when factual content is unchanged, the loop can be manipulated into pathological decision patterns [147, 144]. Such results are a reminder that planning mechanisms can be brittle under targeted perturbations and that evaluation protocols should include robustness checks beyond nominal task success [147].

Finally, planning loops are often evaluated in settings that implicitly assume structured knowledge and controllable environments; these assumptions can break in physically grounded or multi-agent collaboration tasks. Empirical studies in physically grounded collaboration expose failure modes that do not appear in purely textual environments, suggesting that planning modules should be stress-tested under interaction and coordination constraints, not only under offline reasoning metrics [26, 23].

One limitation is that planning comparisons are frequently confounded by unreported protocol details (budgets, tool access, verification), which forces any synthesis to weaken claims rather than guess missing context [38, 44]. Another is that planning is rarely separable from memory and retrieval in long-horizon loops; understanding what is retrieved, when, and under what trust model is essential for interpreting planning results, motivating the next subsection's focus on memory and retrieval mechanisms [29, 110].

Planning determines which decisions to make, whereas memory determines what evidence those decisions can condition on over long horizons.

4.2 Memory and retrieval (RAG)

Seen through the lens of long-horizon agency, memory is not an optional enhancement but a constraint that reshapes what an agent can reliably do. Finite context windows force agents to decide what to retain, retrieve, and trust, and those decisions can dominate behavior under realistic interaction horizons [152, 29, 126, 143]. Retrieval policies and trust assumptions therefore determine what evidence the loop can condition on and what failures are diagnosable, so memory mechanisms should be compared under protocols that make those assumptions visible [38, 44, 73].

Memory mechanisms range from retrieval-augmented prompting to persistent stores that act as long-lived state. Retrieval can be treated as an explicit action (retrieve, reflect, or answer routing), which makes memory access part of the agent loop, while persistent memory can implicitly bias future decisions even when not surfaced as an explicit action [29, 127, 60, 74]. These choices change failure recovery: explicit retrieval can be audited and revised, whereas implicit memory can create silent drift unless interfaces expose provenance and trust signals [152, 111, 132].

Concrete systems illustrate how memory control can be operationalized. MemR^3 treats memory retrieval as an autonomous component and introduces a router that chooses among retrieve, reflect, and answer actions, along with an evidence-gap signal intended to optimize answer quality under constraints [29]. This design suggests that memory is best viewed as a policy: when to retrieve, what to retrieve, and how to revise or discard memories as the loop unfolds [29, 97].

Interactive benchmarks show why memory cannot be evaluated in isolation. In ReAct-style loops, decisions are conditioned on a growing interaction trace, and a small number of in-context examples can yield large gains on interactive environments, but only within the specific

benchmark protocols that define observation and action semantics [127]. As environments shift, the same memory strategy can produce cascading errors, where an early mistake compounds through subsequent decisions [152, 38].

A useful contrast is between memory systems that prioritize breadth of context and those that prioritize structured, verified evidence. Some work focuses on retrieving larger quantities of potentially relevant context, whereas other work stresses retrieval that is auditable and compatible with downstream verification, making the memory interface itself part of the evaluation protocol [42, 111, 103]. In contrast to breadth-first retrieval, evidence-first memory designs treat retrieval as a verifiable action with explicit trust signals, which changes both the error surface and the cost profile of the loop [38, 129].

Error-taxonomy and failure-trajectory datasets make these issues more concrete. AgentErrorTaxonomy and AgentErrorBench classify failures across memory, reflection, planning, and action, and ground error analysis in annotated trajectories drawn from interactive settings such as ALFWorld, GAIA, and WebShop [152, 21]. These resources support protocol-aware comparison by making failure types explicit rather than collapsing them into a single success metric [152, 44].

Security concerns are especially salient for memory. If memory is treated as a writable store that future decisions trust, an adversary can inject benign-looking records that later manipulate behavior, turning memory into a persistent attack channel [110]. This risk is easy to miss unless the threat model and memory interface are explicitly part of the evaluation protocol (what is writable, what is trusted, and how provenance is exposed) [110, 152].

Defenses therefore need to be evaluated as loop components. A-MemGuard reports large reductions in attack success rates (over 95%) with minimal utility cost, illustrating that memory security can be studied empirically when the evaluation protocol includes adversarial interactions and clear success and failure definitions [110]. The broader implication is that memory mechanisms should be compared not only for nominal task success, but also for robustness under realistic adversarial and distribution-shift conditions [110, 152].

Across recent surveys and component taxonomies, memory and retrieval are increasingly treated as core axes for agents, but the evidence base remains uneven when protocols are underspecified. Survey-style synthesis therefore benefits from combining broad taxonomies with explicit protocol anchors (task family, metric, budget, and tool and memory access assumptions) before drawing cross-paper conclusions [33, 28, 38, 75]. Without this alignment, cross-paper numeric comparisons of retrieval systems remain fragile and should be described as protocol-contingent rather than universal [44].

One limitation is that many papers report memory mechanisms without exposing enough provenance or trust signals to support reproducible comparison, which forces synthesis to focus on qualitative trade-offs [38, 111]. Another is that memory is entangled with security and governance: the same mechanism that enables long-horizon competence can create persistent attack channels, so future benchmarks must evaluate memory under threat models, not only under nominal success metrics [110, 152].

5 Learning, Adaptation & Coordination

Adaptation mechanisms change an agent's behavior based on feedback, shifting the focus from fixed policies to self-updating loops. Such updates can occur through prompt and context optimization and revision cycles, but they raise stability questions when evaluation protocols are underspecified [140, 115].

Coordination extends these ideas to multi-agent settings, where role protocols and aggregation rules become part of the method. The subsections in this chapter contrast self-improvement loops with multi-agent coordination patterns, emphasizing how interaction budgets and protocol design determine whether coordination improves robustness or amplifies failure modes [80, 47].

5.1 Self-improvement and adaptation

For system builders, the key decision in self-improvement is how much the agent is allowed to change under feedback. Adaptation can make agents more effective in specific environments, but it can also create instability and obscure what is being evaluated, since the policy at the end of an episode is not the policy at the beginning [102, 99, 6]. Update budgets and feedback channels largely determine whether adaptation gains are stable and transferable, so adaptation mechanisms should be treated as part of the evaluation protocol: improvements are interpretable only relative to what information is used for learning, when updates occur, and which constraints (budget, access, threat model) apply [102, 99, 83].

Self-improvement and adaptation mechanisms include prompt and context optimization, reflection and revision loops, and learning procedures that update policies before or during deployment. These mechanisms are attractive in long-horizon settings because they can correct recurring failure modes without requiring full retraining, but they also introduce new failure modes such as reward hacking and benchmark overfitting [102, 8, 142, 116]. As a result, adaptation claims must be anchored in explicit evaluation settings, including what feedback is available and how much adaptation is allowed [102, 99].

Context optimization provides a concrete, protocol-aware form of adaptation. ACE reports optimizing contexts both offline (e.g., system prompts) and online (e.g., agent memory), with measured gains across agent and domain-specific benchmarks (e.g., +10.6% on agent benchmarks and +8.6% on finance) while reducing adaptation latency and rollout costs under its evaluation setup [140, 69]. Such results are best interpreted as improvements under a specific adaptation protocol, rather than as unconditional superiority of a particular loop design [140, 102].

Self-challenging and self-generated training data highlight a different trade-off. In contrast to one-shot prompt tuning, a self-challenging framework reports over a two-fold improvement for an open model (Llama-3.1-8B-Instruct) on multi-turn tool-use benchmarks using only self-generated training data, suggesting that agents can improve through internal feedback loops when the benchmark protocol supports iterative refinement [149, 12]. The caveat is that such gains may be fragile when the evaluation distribution shifts or when the internal critic aligns to benchmark-specific artifacts [102, 8].

More general learning frameworks attempt to expand beyond prompt-level tuning by enabling agents to learn and evolve before and during test time, incorporating environment-relevant knowledge for planning and improved cooperation [61, 120]. This style of adaptation blurs the line between "agent design" and "training procedure": what appears as a stronger agent may be a stronger learning protocol under the same interface constraints [61, 102].

A useful contrast is between adaptation that primarily changes the agent's context and adaptation that changes the agent's behavior policy more directly. Context-centered approaches can be faster and cheaper but can be brittle to prompt injection and context drift, whereas more direct learning-based approaches can be more stable but require careful accounting of data, feedback, and compute budgets [100, 50, 89]. In either case, evaluation should report adaptation budgets and update frequency, since these determine whether the method scales in realistic deployments [102, 99].

Evaluation protocols for adaptation must also address what is being optimized. Surveys of foundation-model limitations emphasize issues such as hallucinations and limited self-assessment, and suggest that adaptation methods can trade short-term benchmark gains for degraded calibration and interpretability if objectives are underspecified [102, 8, 7]. This motivates conservative synthesis: distinguish improvements in task success from improvements in reliability under uncertainty, and avoid collapsing them into a single score [102, 99].

Security is an especially important axis for self-improvement. Adaptation mechanisms can amplify vulnerabilities by internalizing adversarial signals or by optimizing toward unsafe tool behaviors, so end-to-end security benchmarks for tool-use pipelines are relevant even when their

primary focus is risk [136]. A practical takeaway is that adaptation should be evaluated under threat models that include adversarial inputs and tool-interface attacks, not only under benign task success [136].

Across the literature, adaptation also interacts with multi-agent settings and system architecture. Agent evolution can change communication policies and coordination dynamics, while architectural constraints (tool protocols, memory interfaces) bound what adaptation can safely modify [112, 16]. For synthesis, this suggests treating adaptation as a layer that sits on top of interface and protocol choices: without stable interfaces, adaptation gains are difficult to attribute and difficult to transfer [49, 102]. Across the literature, adaptation also interacts with multi-agent settings and system architecture. Agent evolution can change communication policies and coordination dynamics, while architectural constraints (tool protocols, memory interfaces) bound what adaptation can safely modify [112, 16]. For synthesis, a practical reading is to treat adaptation as a layer that sits on top of interface and protocol choices: without stable interfaces, adaptation gains are difficult to attribute and difficult to transfer [49, 102].

One limitation is that many adaptation papers report gains without sufficiently specifying update budgets and feedback channels, which makes cross-paper comparison fragile and encourages overclaiming [102, 99]. Another is that adaptation introduces governance questions: agents that learn online can change risk profiles after deployment, so evaluations should include both capability and safety metrics under realistic constraints [136, 102].

As agents adapt over time, multi-agent systems introduce an additional dimension of change: coordination protocols that distribute decision-making and verification across roles.

5.2 Multi-agent coordination

A sharp contrast in agent design is between single-agent loops that internalize all decisions, whereas multi-agent systems distribute reasoning, tool use, and verification across specialized roles. Distribution can improve coverage and robustness, but it also introduces coordination overhead and new failure modes, so evaluation must specify interaction budgets and aggregation protocols rather than reporting a single success number [47, 3, 123, 122]. Role protocols and cost models largely determine whether coordination gains are comparable across papers, so multi-agent results are only comparable when communication channels and accounting assumptions are explicit [47, 3, 124, 88].

Multi-agent coordination mechanisms include role specialization (planner, verifier, executor), communication protocols (messages, shared memory, structured schemas), and aggregation rules (vote, debate, referee). These choices reshape the effective action space: coordination can add verification capacity but can also amplify cascading failures when agents reinforce each other's errors [15, 5, 57, 118, 133]. As a result, "better coordination" should be stated as a protocol-contingent claim tied to interaction budgets and observable artifacts (e.g., what evidence verifiers can access) [3, 18, 128].

Rubric-based evaluation provides one way to make coordination protocols explicit. ARCANE is evaluated on tasks requiring multi-step reasoning and tool use using a corpus of 219 labeled rubrics derived from a benchmark, which helps separate coordination quality from superficial success metrics [80, 101]. This style of evaluation is valuable because it makes the grading protocol legible, enabling more meaningful comparisons of role design and aggregation rules [80].

Coordination is also tested in collaborative settings where agents must share observations, negotiate roles, and recover from each other's errors. Such settings illustrate that protocol details (communication latency, observation sharing, and interaction budgets) can dominate outcomes, so coordination mechanisms should be compared under matched observability and communication constraints, not only under nominal task labels [61, 47, 17].

Domain-specific multi-agent systems further illustrate how protocol assumptions become part of the method. In a drug-discovery setting, a multi-agent approach reports large improve-

ments over a non-reasoning multi-agent baseline (e.g., a 45% F1 gain on a kinase inhibitor dataset), suggesting that coordination can matter even when the underlying model family is similar [43]. The transferable lesson is not the number itself, but the requirement that datasets, metrics, and interaction protocols be explicit before coordination claims are generalized [43, 18].

Evaluation settings can also involve multi-agent scoring and assessment. AutoSCORE is evaluated on multiple datasets from the ASAP benchmark with both proprietary and open-source models, illustrating that the evaluation protocol may include grading and judgment components rather than purely environment success [107]. This makes coordination questions salient: when agents include judges or critics, the protocol must specify how judgments are aggregated and what signals are available to avoid hidden leakage or evaluator drift [107, 80].

Security verification offers a complementary coordination pattern: centralized supervision with delegated execution. MARVEL uses a supervisor agent that derives security policies and delegates validation tasks to executor agents, reflecting a common architectural motif where coordination is organized around a central planner with specialized workers [24]. Such hierarchies can improve tractability, but they also concentrate failure modes in the supervisor's policy and require explicit accounting of what documentation and evidence each role can access [24, 18].

A useful architectural contrast is centralized supervision versus more decentralized coordination. Centralized designs can simplify protocol design and reduce message overhead; in contrast, decentralized designs can increase redundancy and verification at the cost of coordination complexity [47, 5]. In both cases, reporting should include interaction budgets and cost and latency constraints, since coordination gains can disappear when communication is expensive or time-limited [3, 5].

Across these studies, multi-agent systems tend to trade raw capability for controllability and verification. Systems that embed critics, referees, or specialized executors can surface errors that a single-agent loop would miss, but they can also create new pathways for error amplification and increased costs if protocols are not carefully bounded [80, 15, 107]. For synthesis, the conservative move is to compare coordination mechanisms within shared protocol envelopes and to treat cross-paper differences in interaction budgets as the primary confounder [3, 47].

One limitation is that coordination methods often face scalability and real-time response constraints, which means that improvements observed under generous interaction budgets may not transfer to deployment settings [3, 5]. Another is that coordination interacts with learning and adaptation: frameworks that allow agents to evolve before or during test time can change communication policies, making it essential to specify what changes are permitted and when [61, 102].

6 Evaluation & Risks

Evaluation protocols determine which claims about agents are interpretable. Benchmark suites span diverse tasks and metrics, but protocol drift (tool access, budgets, environment versions, reporting) can dominate outcomes and undermine synthesis when assumptions are implicit [37, 137].

Risk surfaces are tightly coupled to these protocols: tool use expands the attack surface, and safety outcomes depend on threat models and interface boundaries. The two subsections in this chapter therefore cover (i) benchmarks and protocol anchors for comparability and (ii) safety, security, and governance considerations that must be reflected in evaluation designs [92, 136].

6.1 Benchmarks and evaluation protocols

A practical decision for evaluating agents is what to treat as "success" and what constraints to enforce while measuring it. Benchmarks vary not only in task families and metrics, but also in budgets, tool access assumptions, and environment stability, so protocols determine whether

results are comparable or merely adjacent anecdotes [11, 40, 92, 52, 13]. In particular, nominally similar tasks can differ in tool permissions, logging granularity, or stopping criteria, which can change both reported success and the distribution of failures [40]. The central takeaway is that evaluation protocols should be written as explicit contracts (task + metric + constraint), and surveys should synthesize evidence primarily within shared contracts rather than across incompatible ones [45, 139].

Benchmark suites increasingly span heterogeneous domains: web interaction, embodied control, tool use, code, and multi-agent coordination. This breadth is valuable for coverage but also creates protocol drift: different tasks implicitly assume different observability, action interfaces, and cost models, which can dominate outcomes for long-horizon loops [11, 137, 134, 86]. As a result, evaluation papers that expose protocol assumptions can be as important for synthesis as new agent architectures [102, 45, 98, 2].

Concrete examples illustrate the sensitivity to model access and infrastructure. DataSciBench reports that API-based models outperform open-source models on its metrics, whereas a strong open model achieves the highest score among open-source baselines under that benchmark's setup [135]. Such results are informative, but only when the evaluation discloses what the API access implies (tooling, cost, latency) and how those constraints affect agent behavior relative to offline models [139, 11].

Cost and latency constraints are themselves evaluation axes. EvoRoute reports that, when integrated into off-the-shelf agentic systems, it can sustain or improve performance while reducing execution cost by up to 80% and latency by over 70% on agentic benchmarks such as GAIA and BrowseComp+ [137]. This underscores a key protocol point: reporting budgets and time constraints is not optional, because "better" agents can be strictly worse once costs are normalized [137, 40].

Security and robustness benchmarks make protocol design even more explicit by including adversarial interactions. Progent reports reducing attack success rates to 0% while preserving utility and speed across multiple agent use cases and benchmarks, illustrating how evaluation can incorporate threat models and mitigation efficacy into the same measurement loop [95]. These settings help prevent a common failure mode in capability evaluation: declaring success on benign tasks while ignoring interface-layer vulnerabilities [136, 32].

Efficiency and long-context constraints are also becoming first-class evaluation targets. ACBench spans multiple tasks, capabilities, and model variants (including compression techniques), emphasizing that evaluation should account for how model size, pruning and quantization, and context management interact with agent loop performance [27]. This aligns with the broader observation that many agent behaviors are constrained by context budgets and memory policies rather than by raw reasoning quality [152, 137].

Simulated environments and agent behavior suites provide another evaluation route. Simulators can stress multi-step interaction patterns and reveal systematic shortcomings in evaluation methods, such as constrained evaluation coverage, benchmark vulnerability, and unobjective metrics [67]. Domain-specific simulators and evaluations can complement broad suites, but they also require careful disclosure of environment assumptions and versioning to avoid irreproducible comparisons [91, 40].

Several surveys of evaluation practice emphasize recurring pitfalls: benchmark leakage, implicit tool access assumptions, and metrics that do not capture failure recovery or safety-critical behavior. These issues make it possible for systems to improve scores without improving robustness, especially when evaluation protocols are underspecified or easily gamed [40, 67, 87]. For synthesis, this motivates using evaluation papers to identify which results should be treated as provisional rather than definitive [139, 11, 119].

Across benchmark suites, protocol papers, and surveys, a consistent theme is that comparability requires explicit protocol anchors. When benchmarks share tasks and metrics, head-to-head comparison is meaningful; whereas when they differ, the correct synthesis is to describe

how protocol choices reshape trade-offs and failure modes rather than declaring a single method superior [92, 45, 102, 151]. This is also where reader-facing tables can help: they compress which protocol constraints matter for each benchmark family, reducing the temptation to treat all scores as commensurate [137, 107].

One limitation is that protocol reporting is still inconsistent, especially for tool access, costs, and environment versions, which forces surveys to weaken claims to avoid guessing missing context [40, 139]. Another is that benchmarks that focus on capability without integrating threat models can misrepresent deployability, motivating the next subsection's focus on safety, security, and governance as integral to evaluation design [136, 95].

Protocol contracts clarify what success means, but they also determine what risks are exercised, so the next step is to examine safety and governance under explicit threat models.

6.2 Safety, security, and governance

From the perspective of deployed agents, safety and security are not add-on concerns; they are consequences of interface choices and protocol assumptions. Tool use expands the attack surface through prompt injection, data exfiltration, and privilege misuse, and tool protocols and permission models can largely determine the effective risk profile even for the same underlying agent loop [136, 32]. Governance therefore requires protocol-aware evaluation: threat models, system boundaries, and monitoring assumptions must be represented in the evaluation contract, not deferred to informal caveats [113, 136].

Threat models for tool-using agents are shaped by the agent-tool communication loop. Attacks can target tool descriptions, tool arguments, retrieval channels, or error-handling logic, exploiting the fact that agents treat tool outputs as part of their observation stream [136, 32]. This implies that "robustness" cannot be assessed without specifying what tools exist, what the agent is allowed to call, and what information is trusted after a call [32, 113].

Comparative evaluations illustrate that protocol choice changes both capability and vulnerability. One analysis reports higher overall attack success rates under function calling than under MCP (73.5% vs 62.59%), while also distinguishing system-centric and LLM-centric exposure patterns under these protocols [32]. Such numbers are only interpretable when the protocol defines attack channels and system boundaries, reinforcing that security results must be read as protocol-contingent rather than universal [32, 136].

Taxonomies make this dependence explicit. MSB contributes an attack taxonomy with a dozen attack types (including name collision and tool-description prompt injection variants) and evaluates resistance throughout the tool-use pipeline, covering planning, invocation, and response handling [136]. This pipeline-level framing is important because failures often arise from interactions among components rather than from a single prompt vulnerability [136, 85].

Defensive systems highlight that mitigation can change both security and system efficiency under realistic workflows. BridgeScope reports enabling agents to operate databases more effectively while reducing token usage by up to 80% through improved security awareness, illustrating that protocol-aware defenses can trade small interface restrictions for large safety and efficiency gains [113]. For synthesis, such results should be tied to the workflow protocol (what queries and tools are allowed, what logging exists) rather than treated as generic "agent safety" improvements [113, 32].

End-to-end defenses can also be benchmarked directly. Progent reports reducing attack success rates to 0% across multiple agent use cases and benchmarks while preserving utility and speed, suggesting that defenses can be evaluated under comparable protocols instead of relying on ad hoc red teaming anecdotes [95]. This style of benchmark-driven security evaluation is essential for governance because it supports reproducible, testable claims about mitigations [95, 136].

Monitoring and cost-aware attacks expose additional governance constraints. Some attacks operate at the tool layer under the guise of correct task completion, inflating token usage,

monetary cost, and energy consumption, which can be as damaging as capability compromise in deployed systems [81, 31]. These failure modes motivate reporting and enforcing budget policies and detection mechanisms as part of the protocol, not as optional operational folklore [9, 31].

In contrast to threat models that focus only on correctness, governance-oriented evaluations often require monitoring and auditing: which actions were taken, under what permissions, and what signals a system used to decide that an interaction remained safe. Monitoring systems and checklists for agent operations provide concrete guidance on how to instrument such protocols so violations are detectable rather than implicit [108, 9].

Governance considerations extend beyond technical mitigations to domain and compliance constraints. Surveys of domain deployments (e.g., agents in specialized professional workflows) emphasize that accountability, auditability, and policy boundaries are often the limiting factors, even when nominal task success is achievable [70, 34]. This pattern indicates that evaluation protocols for governance should include traceability and controllability metrics, not just task success [85, 139].

Across security benchmarks and defensive systems, a consistent synthesis is that risk is interface-dependent. Protocols that expose more powerful actions without corresponding observability and guardrails can increase both capability and vulnerability, whereas stricter protocols can improve reliability at the cost of expressivity [32, 136, 113]. As a result, governance arguments should be framed as design trade-offs that depend on explicit threat models and constraints, not as generic prescriptions for "safe agents" [136, 85].

One limitation is that many security evaluations are still fragmented across incompatible benchmarks and tool stacks, making it difficult to aggregate evidence without strong protocol alignment [136, 32]. Another is that governance requires coupling security evaluation with the broader evaluation ecosystem: defenses must be assessed under the same budgets, tool access assumptions, and environment constraints as capability claims, or else improvements will not transfer to deployment [95, 113].

7 Discussion

One cross-cutting implication is that many "agent improvements" are inseparable from interface assumptions. Changing a tool protocol, permission model, or observation channel can shift not only success rates but also which failure modes are even observable, so comparisons that omit these details risk overclaiming generality [138, 22, 53, 154]. A practical reading is to treat interface contracts as part of the method, not as a peripheral implementation detail [77, 72].

A second implication concerns evaluation drift. Benchmark suites cover increasingly diverse domains, but task, metric, and budget choices vary across papers, and even small protocol differences can dominate outcomes for long-horizon loops [37, 92, 137]. Where protocol context is missing, the safest synthesis is comparative but conservative: emphasize patterns of trade-offs and failure modes, and downgrade claims that depend on unreported budgets or tool access [38, 44, 139].

Risk surfaces also scale with capability. Tool use expands the attack surface through prompt injection and unintended tool behaviors, and mitigation often depends on the same interface decisions that govern capability and evaluation [136, 32, 110]. This suggests that safety and evaluation should be co-designed: threat models and governance constraints need to be represented in the evaluation protocol, not deferred to a separate checklist [113, 154].

Looking forward, progress is likely to depend on making protocols more explicit and more comparable: standardizing what is reported (tool access, budgets, environment versions), supporting ablation-friendly evaluations for loop components (planning vs memory vs orchestration), and documenting failure modes in a way that enables cross-paper synthesis [37, 135, 25]. These practices are not mere hygiene; they determine whether evidence accumulates across

studies or remains siloed by incompatible assumptions [139, 28].

8 Conclusion

LLM agents can be understood as closed-loop systems whose apparent capabilities depend as much on interface contracts and protocol constraints as on model quality [127, 78, 146]. Reading the literature through this lens clarifies why results often fail to transfer: tool access, budgets, and environment assumptions change the meaning of "success" and the visibility of failure modes [37, 92, 137].

The synthesis in this paper emphasizes protocol-aware contrasts across four lenses: foundations and interfaces, planning and memory components, adaptation and coordination mechanisms, and evaluation and risks. Across these lenses, the most reusable takeaways are rarely single system designs; they are patterns of trade-offs and failure modes that persist across benchmarks when assumptions are made explicit [38, 44, 139].

A final implication is that evaluation and governance are coupled for tool-using agents. Security and safety outcomes hinge on the same interface decisions that drive capability, so threat models and protocol constraints should be treated as first-class evaluation objects rather than post hoc caveats [136, 32, 110].

A Tables

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] R. M. Aratchige and W. M. K. S. Ilmini. Llms working in harmony: A survey on the technological aspects of building effective llm-based multi agent systems. *arXiv preprint arXiv:2504.01963v1*, 2025. URL http://arxiv.org/abs/2504.01963v1.

[4] David Bai, Ishika Singh, David Traum, and Jesse Thomason. Twostep: Multi-agent task planning using classical planners and large language models. *arXiv preprint arXiv:2403.17246v2*, 2024. URL http://arxiv.org/abs/2403.17246v2.

[5] Jonas Becker, Lars Benedikt Kaesberg, Niklas Bauer, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Mallm: Multi-agent large language models framework. *arXiv preprint arXiv:2509.11656v3*, 2025. URL http://arxiv.org/abs/2509.11656v3.

[6] 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.

[7] Manasa Bharadwaj, Nikhil Verma, and Kevin Ferreira. Omnireflect: Discovering transferable constitutions for llm agents via neuro-symbolic reflections. *arXiv preprint arXiv:2506.17449v1*, 2025. URL http://arxiv.org/abs/2506.17449v1.

Table 1: Representative agent approaches and their interface and loop assumptions (selected examples).

Approach / work family	Core idea	Interface / loop assumption	Key refs
Reasoning-action interleaving	interleave reasoning traces with environment and tool actions	closed-loop, multi-step interaction where actions become context for later decisions	[127]
Memory retrieval as an agent module	treat memory retrieval as an autonomous component	retrieval decisions are part of the agent loop; memory access policy affects comparability	[29]
Interface hardening (memory and tool guardrails)	reduce attack success by constraining memory and tool interactions	adversarial inputs are in-scope; guardrails are evaluated as part of the loop	[110]
Tool protocol orchestration	compare tool modes under a protocolized interface	explicit tool protocol; repeated interactions surface reliability limits	[77]
Benchmark-driven agent framework	evaluate agents across diverse settings (web, embodied, tool use, games)	comparability depends on consistent environments and tool-access assumptions	[92]
Benchmark landscape synthesis	aggregate large benchmark landscapes to track evaluation drift	protocols vary by task, metric, and budget; alignment is needed for cross-paper comparison	[37]
Context optimization for agents	optimize contexts offline and online as an adaptation mechanism	prompt and context are part of controllable state; stability depends on protocol constraints	[140]
Security benchmarking for tool protocols	characterize attack surfaces and vulnerabilities in tool interfaces	threat model and system and tool boundary determine what "attack success" means	[136, 32]

[8] Ahsan Bilal, Muhammad Ahmed Mohsin, Muhammad Umer, Muhammad Awais Khan Bangash, and Muhammad Ali Jamshed. Meta-thinking in llms via multi-agent reinforcement learning: A survey. *arXiv preprint arXiv:2504.14520v1*, 2025. URL http://arxiv.org/abs/2504.14520v1.

[9] 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.

[10] Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683v1*, 2025. URL http://arxiv.org/abs/2505.19683v1.

[11] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109v9*, 2023. URL http://arxiv.org/abs/2307.03109v9.

[12] Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Hei-

Table 2: Evaluation settings and protocol anchors for LLM agents (examples).

Benchmark / setting	Task family + metric (example)	Key protocol constraints (what must be reported)	Key refs
ALFWorld; WebShop	interactive decision making; task success	step budget; tool access; environment stochasticity; cost and latency	[127]
ScaleMCP	tool-protocol interactions; success over repeated sessions	consecutive user interactions; model suite; protocol versioning; tool availability	[77]
MCP Security Benchmark (MSB)	attacks on tool protocols; attack success rate	threat model; injection channel; system and tool boundary; mitigations	[136, 32]
GAIA; BrowseComp+	agentic browsing and composite tasks; success rate	tool budget; time and step limits; evaluation rubric; access constraints	[137]
ASAP benchmark (AutoSCORE)	scoring tasks; accuracy and quality metrics	model choice (open vs proprietary); prompt format; dataset splits	[107]
Rubric-based multi-agent evaluation (ARCANE)	multi-agent tasks; rubric-based scoring	role protocol; aggregation and voting; interaction budget	[80]

necke, 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.

[13] 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.

[14] Jun Yu Chen and Tao Gao. Apt: Architectural planning and text-to-blueprint construction using large language models for open-world agents. *arXiv preprint arXiv:2411.17255v2*, 2024. URL http://arxiv.org/abs/2411.17255v2.

[15] Junzhe Chen, Xuming Hu, Shuodi Liu, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Lijie Wen. Llmarena: Assessing capabilities of large language models in dynamic multi-agent environments. *arXiv preprint arXiv:2402.16499v1*, 2024. URL http://arxiv.org/abs/2402.16499v1.

[16] Ke Chen, Peiran Wang, Yaoning Yu, Xianyang Zhan, and Haohan Wang. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744v2*, 2025. URL http://arxiv.org/abs/2508.02744v2.

[17] Weizhe Chen, Sven Koenig, and Bistra Dilkina. Why solving multi-agent path finding with large language model has not succeeded yet. *arXiv preprint arXiv:2401.03630v2*, 2024. URL http://arxiv.org/abs/2401.03630v2.

[18] Yiye Chen, Harpreet Sawhney, Nicholas Gydé, Yanan Jian, Jack Saunders, Patricio Vela, and Ben Lundell. Schema-guided scene-graph reasoning based on multi-agent large language model system. *arXiv preprint arXiv:2502.03450v2*, 2025. URL http://arxiv.org/abs/2502.03450v2.

[19] Yongchao Chen, Harsh Jhamtani, Srinagesh Sharma, Chuchu Fan, and Chi Wang. Steering large language models between code execution and textual reasoning. *arXiv preprint arXiv:2410.03524v2*, 2024. URL http://arxiv.org/abs/2410.03524v2.

[20] 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.

[21] Yuan Chiang, Elvis Hsieh, Chia-Hong Chou, and Janosh Riebesell. Llamp: Large language model made powerful for high-fidelity materials knowledge retrieval and distillation. *arXiv preprint arXiv:2401.17244v3*, 2024. URL http://arxiv.org/abs/2401.17244v3.

[22] 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.

[23] Kun Chu, Xufeng Zhao, Cornelius Weber, and Stefan Wermter. Llm+map: Bimanual robot task planning using large language models and planning domain definition language. *arXiv preprint arXiv:2503.17309v1*, 2025. URL http://arxiv.org/abs/2503.17309v1.

[24] Luca Collini, Baleegh Ahmad, Joey Ah-kiow, and Ramesh Karri. Marvel: Multi-agent rtl vulnerability extraction using large language models. *arXiv preprint arXiv:2505.11963v2*, 2025. URL http://arxiv.org/abs/2505.11963v2.

[25] 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.

[26] 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.

[27] Peijie Dong, Zhenheng Tang, Xiang Liu, Lujun Li, Xiaowen Chu, and Bo Li. Can compressed llms truly act? an empirical evaluation of agentic capabilities in llm compression. *arXiv preprint arXiv:2505.19433v2*, 2025. URL http://arxiv.org/abs/2505.19433v2.

[28] 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.

[29] Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. Memr^3: Memory retrieval via reflective reasoning for llm agents. *arXiv preprint arXiv:2512.20237v1*, 2025. URL http://arxiv.org/abs/2512.20237v1.

[30] 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.

[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] Zhuohan Ge, Darian Li, Yubo Wang, Nicole Hu, Xinyi Zhu, Haoyang Li, Xin Zhang, Mingtao Zhang, Shihao Qi, Yuming Xu, Han Shi, Chen Jason Zhang, and Qing Li. Survey and experiments on mental disorder detection via social media: From large language models and rag to agents. *arXiv preprint arXiv:2504.02800v4*, 2025. URL http://arxiv.org/abs/2504.02800v4.

[34] Jindong Han, Yansong Ning, Zirui Yuan, Hang Ni, Fan Liu, Tengfei Lyu, and Hao Liu. Large language model powered intelligent urban agents: Concepts, capabilities, and applications. *arXiv preprint arXiv:2507.00914v1*, 2025. URL http://arxiv.org/abs/2507.00914v1.

[35] 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.

[36] 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.

[37] Ming Hu, Chenglong Ma, Wei Li, Wanghan Xu, Jiamin Wu, Jucheng Hu, Tianbin Li, Guohang Zhuang, Jiaqi Liu, Yingzhou Lu, Ying Chen, Chaoyang Zhang, Cheng Tan, Jie Ying, Guocheng Wu, Shujian Gao, Pengcheng Chen, Jiashi Lin, Haitao Wu, Lulu Chen, Fengxiang Wang, Yuanyuan Zhang, Xiangyu Zhao, Feilong Tang, Encheng Su, Junzhi Ning, Xinyao Liu, Ye Du, Changkai Ji, Pengfei Jiang, Cheng Tang, Ziyan Huang, Jiyao Liu, Jiaqi Wei, Yuejin Yang, Xiang Zhang, Guangshuai Wang, Yue Yang, Huihui Xu, Ziyang Chen, Yizhou Wang, Chen Tang, Jianyu Wu, Yuchen Ren, Siyuan Yan, Zhonghua Wang, Zhongxing Xu, Shiyan Su, Shangquan Sun, Runkai Zhao, Zhisheng Zhang, Dingkang Yang, Jinjie Wei, Jiaqi Wang, Jiahao Xu, Jiangtao Yan, Wenhao Tang, Hongze Zhu, Yu Liu, Fudi Wang, Yiqing Shen, Yuanfeng Ji, Yanzhou Su, Tong Xie, Hongming Shan, Chun-Mei Feng, Zhi Hou, Diping Song, Lihao Liu, Yanyan Huang, Lequan Yu, Bin Fu, Shujun Wang, Xiaomeng Li, Xiaowei Hu, Yun Gu, Ben Fei, Benyou Wang, Yuewen Cao, Minjie Shen, Jie Xu, Haodong Duan, Fang Yan, Hongxia Hao, Jielan Li, Jiajun Du, Yanbo Wang, Imran Razzak, Zhongying Deng, Chi Zhang, Lijun Wu, Conghui He, Zhaohui Lu, Jinhai Huang, Wenqi Shao, Yihao Liu, Siqi Luo, Yi Xin, Xiaohong Liu, Fenghua Ling, Yuqiang Li, Aoran Wang, Siqi Sun, Qihao Zheng, Nanqing Dong, Tianfan Fu, Dongzhan Zhou, Yan Lu, Wenlong Zhang, Jin Ye, Jianfei Cai, Yirong Chen, Wanli Ouyang, Yu Qiao, Zongyuan Ge, Shixiang Tang, Junjun He, Chunfeng Song, Lei Bai, and Bowen Zhou. A survey of scientific large language models: From data foundations to agent frontiers. *arXiv preprint arXiv:2508.21148v2*, 2025. URL http://arxiv.org/abs/2508.21148v2.

[38] Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257v2*, 2025. URL http://arxiv.org/abs/2507.05257v2.

[39] 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.

[40] Jiaxing Huang and Jingyi Zhang. A survey on evaluation of multimodal large language models. *arXiv preprint arXiv:2408.15769v1*, 2024. URL http://arxiv.org/abs/2408.15769v1.

[41] Jiayuan Huang, Runlong He, Danyal Zaman Khan, Evangelos B. Mazomenos, Danail Stoyanov, Hani Marcus, Linzhe Jiang, Matthew J Clarkson, and Mobarak I. Hoque. Surgical ai copilot: Energy-based fourier gradient low-rank adaptation for surgical llm agent reasoning and planning. *arXiv preprint arXiv:2503.09474v2*, 2025. URL http://arxiv.org/abs/2503.09474v2.

[42] 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.

[43] Yoshitaka Inoue, Tianci Song, Xinling Wang, Augustin Luna, and Tianfan Fu. Drugagent: Multi-agent large language model-based reasoning for drug-target interaction prediction. *arXiv preprint arXiv:2408.13378v4*, 2024. URL http://arxiv.org/abs/2408.13378v4.

[44] 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.

[45] 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.

[46] 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.

[47] Feibo Jiang, Li Dong, Yubo Peng, Kezhi Wang, Kun Yang, Cunhua Pan, Dusit Niyato, and Octavia A. Dobre. Large language model enhanced multi-agent systems for 6g communications. *arXiv preprint arXiv:2312.07850v1*, 2023. URL http://arxiv.org/abs/2312.07850v1.

[48] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163v1*, 2024. URL http://arxiv.org/abs/2402.11163v1.

[49] Zhonghao Jiang, David Lo, and Zhongxin Liu. Agentic software issue resolution with large language models: A survey. *arXiv preprint arXiv:2512.22256v1*, 2025. URL http://arxiv.org/abs/2512.22256v1.

[50] Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen. From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479v2*, 2024. URL http://arxiv.org/abs/2408.02479v2.

[51] Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling llm agent into small models with retrieval and code tools. *arXiv preprint arXiv:2505.17612v2*, 2025. URL http://arxiv.org/abs/2505.17612v2.

[52] 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.

[53] 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.

[54] Andrew Kiruluta. A novel architecture for symbolic reasoning with decision trees and llm agents. *arXiv preprint arXiv:2508.05311v1*, 2025. URL http://arxiv.org/abs/2508.05311v1.

[55] 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.

[56] Siqi Lai, Yansong Ning, Zirui Yuan, Zhixi Chen, and Hao Liu. Ustbench: Benchmarking and dissecting spatiotemporal reasoning of llms as urban agents. *arXiv preprint arXiv:2505.17572v1*, 2025. URL http://arxiv.org/abs/2505.17572v1.

[57] Hao Duong Le, Xin Xia, and Zhang Chen. Multi-agent causal discovery using large language models. *arXiv preprint arXiv:2407.15073v3*, 2024. URL http://arxiv.org/abs/2407.15073v3.

[58] Chenliang Li, Hehong Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, Hongzhu Shi, Ji Zhang, Fei Huang, and Jingren Zhou. Modelscope-agent: Building your customizable agent system with open-source large language models. *arXiv preprint arXiv:2309.00986v1*, 2023. URL http://arxiv.org/abs/2309.00986v1.

[59] Chuanhao Li, Runhan Yang, Tiankai Li, Milad Bafarassat, Kourosh Sharifi, Dirk Bergemann, and Zhuoran Yang. Stride: A tool-assisted llm agent framework for strategic and interactive decision-making. *arXiv preprint arXiv:2405.16376v2*, 2024. URL http://arxiv.org/abs/2405.16376v2.

[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] 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.

[62] 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.

[63] 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.

[64] 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.

[65] Yuxuan Li, Sauvik Das, and Hirokazu Shirado. What makes llm agent simulations useful for policy? insights from an iterative design engagement in emergency preparedness. *arXiv preprint arXiv:2509.21868v1*, 2025. URL http://arxiv.org/abs/2509.21868v1.

[66] Zichuan Li, Jian Cui, Xiaojing Liao, and Luyi Xing. Les dissonances: Cross-tool harvesting and polluting in pool-of-tools empowered llm agents. *arXiv preprint arXiv:2504.03111v3*, 2025. URL http://arxiv.org/abs/2504.03111v3.

[67] Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. Agentsims: An open-source sandbox for large language model evaluation. *arXiv preprint arXiv:2308.04026v1*, 2023. URL http://arxiv.org/abs/2308.04026v1.

[68] Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management. *arXiv preprint arXiv:2508.21433v3*, 2025. URL http://arxiv.org/abs/2508.21433v3.

[69] Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, Xiaoyu Liang, WenHao Wang, Tianze Wu, Zhengxi Lu, Siheng Chen, LiLinghao, Hao Wang, Guanjing Xiong, Yong Liu, and Hongsheng Li. Llm-powered gui agents in phone automation: Surveying progress and prospects. *arXiv preprint arXiv:2504.19838v3*, 2025. URL http://arxiv.org/abs/2504.19838v3.

[70] 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.

[71] Tianming Liu, Jirong Yang, Yafeng Yin, Manzi Li, Linghao Wang, and Zheng Zhu. Aligning llm agents with human learning and adjustment behavior: a dual agent approach. *arXiv preprint arXiv:2511.00993v1*, 2025. URL http://arxiv.org/abs/2511.00993v1.

[72] 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.

[73] WenTao Liu, Ruohua Zhang, Aimin Zhou, Feng Gao, and JiaLi Liu. Echo: A large language model with temporal episodic memory. *arXiv preprint arXiv:2502.16090v1*, 2025. URL http://arxiv.org/abs/2502.16090v1.

[74] Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382v3*, 2023. URL http://arxiv.org/abs/2309.17382v3.

[75] Junru Lu, Jiarui Qin, Lingfeng Qiao, Yinghui Li, Xinyi Dai, Bo Ke, Jianfeng He, Ruizhi Qiao, Di Yin, Xing Sun, Yunsheng Wu, Yinsong Liu, Shuangyin Liu, Mingkong Tang, Haodong Lin, Jiayi Kuang, Fanxu Meng, Xiaojuan Tang, Yunjia Xi, Junjie Huang, Haotong Yang, Zhenyi Shen, Yangning Li, Qianwen Zhang, Yifei Yu, Siyu An, Junnan Dong, Qiufeng Wang, Jie Wang, Keyu Chen, Wei Wen, Taian Guo, Zhifeng Shen, Daohai Yu, Jiahao Li, Ke Li, Zongyi Li, and Xiaoyu Tan. Youtu-llm: Unlocking the native agentic potential for lightweight large language models. *arXiv preprint arXiv:2512.24618v2*, 2025. URL http://arxiv.org/abs/2512.24618v2.

[76] Keer Lu, Chong Chen, Xili Wang, Bin Cui, Yunhuai Liu, and Wentao Zhang. Pilotrl: Training language model agents via global planning-guided progressive reinforcement learning. *arXiv preprint arXiv:2508.00344v4*, 2025. URL http://arxiv.org/abs/2508.00344v4.

[77] 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.

[78] 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.

[79] David Maranto. Llmsat: A large language model-based goal-oriented agent for autonomous space exploration. *arXiv preprint arXiv:2405.01392v1*, 2024. URL http://arxiv.org/abs/2405.01392v1.

[80] Charlie Masters, Marta Grześkiewicz, and Stefano V. Albrecht. Arcane: A multi-agent framework for interpretable and configurable alignment. *arXiv preprint arXiv:2512.06196v1*, 2025. URL http://arxiv.org/abs/2512.06196v1.

[81] 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.

[82] Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. Malt: Improving reasoning with multi-agent llm training. *arXiv preprint arXiv:2412.01928v3*, 2024. URL http://arxiv.org/abs/2412.01928v3.

[83] Xinyi Mou, Xuanwen Ding, Qi He, Liang Wang, Jingcong Liang, Xinnong Zhang, Libo Sun, Jiayu Lin, Jie Zhou, Xuanjing Huang, and Zhongyu Wei. From individual to society: A survey on social simulation driven by large language model-based agents. *arXiv preprint arXiv:2412.03563v1*, 2024. URL http://arxiv.org/abs/2412.03563v1.

[84] Humza Nusrat, Luke Francisco, Bing Luo, Hassan Bagher-Ebadian, Joshua Kim, Karen Chin-Snyder, Salim Siddiqui, Mira Shah, Eric Mellon, Mohammad Ghassemi, Anthony Doemer, Benjamin Movsas, and Kundan Thind. Automated stereotactic radiosurgery planning using a human-in-the-loop reasoning large language model agent. *arXiv preprint arXiv:2512.20586v1*, 2025. URL http://arxiv.org/abs/2512.20586v1.

[85] 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.

[86] Biqing Qi, Kaiyan Zhang, Kai Tian, Haoxiang Li, Zhang-Ren Chen, Sihang Zeng, Ermo Hua, Hu Jinfang, and Bowen Zhou. Large language models as biomedical hypothesis generators: A comprehensive evaluation. *arXiv preprint arXiv:2407.08940v2*, 2024. URL http://arxiv.org/abs/2407.08940v2.

[87] Subhey Sadi Rahman, Md. Adnanul Islam, Md. Mahbub Alam, Musarrat Zeba, Md. Abdur Rahman, Sadia Sultana Chowa, Mohaimenul Azam Khan Raiaan, and Sami Azam.

Hallucination to truth: A review of fact-checking and factuality evaluation in large language models. *arXiv preprint arXiv:2508.03860v2*, 2025. URL http://arxiv.org/abs/2508.03860v2.

[88] Pouria Rouzrokh, Bardia Khosravi, Parsa Rouzrokh, and Moein Shariatnia. Lattereview: A multi-agent framework for systematic review automation using large language models. *arXiv preprint arXiv:2501.05468v2*, 2025. URL http://arxiv.org/abs/2501.05468v2.

[89] 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.

[90] Colin Samplawski, Adam D. Cobb, and Susmit Jha. Agent: An aerial vehicle generation and design tool using large language models. *arXiv preprint arXiv:2504.08981v1*, 2025. URL http://arxiv.org/abs/2504.08981v1.

[91] Gyuhyeon Seo, Jungwoo Yang, Junseong Pyo, Nalim Kim, Jonggeun Lee, and Yohan Jo. Simuhome: A temporal- and environment-aware benchmark for smart home llm agents. *arXiv preprint arXiv:2509.24282v2*, 2025. URL http://arxiv.org/abs/2509.24282v2.

[92] 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.

[93] Chen Shen, Wanqing Zhang, Kehan Li, Erwen Huang, Haitao Bi, Aiying Fan, Yiwen Shen, Hongmei Dong, Ji Zhang, Yuming Shao, Zengjia Liu, Xinshe Liu, Tao Li, Chunxia Yan, Shuanliang Fan, Di Wu, Jianhua Ma, Bin Cong, Zhenyuan Wang, and Chunfeng Lian. Feat: A multi-agent forensic ai system with domain-adapted large language model for automated cause-of-death analysis. *arXiv preprint arXiv:2508.07950v1*, 2025. URL http://arxiv.org/abs/2508.07950v1.

[94] 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.

[95] 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.

[96] 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.

[97] Jia Ao Sun, Hao Yu, Fabrizio Gotti, Fengran Mo, Yihong Wu, Yuchen Hui, and Jian-Yun Nie. Search-on-graph: Iterative informed navigation for large language model reasoning on knowledge graphs. *arXiv preprint arXiv:2510.08825v1*, 2025. URL http://arxiv.org/abs/2510.08825v1.

[98] Yihong Tang, Kehai Chen, Liang Yue, Jinxin Fan, Caishen Zhou, Xiaoguang Li, Yuyang Zhang, Mingming Zhao, Shixiong Kai, Kaiyang Guo, Xingshan Zeng, Wenjing Cun, Lifeng Shang, and Min Zhang. Empowering real-world: A survey on the technology, practice, and evaluation of llm-driven industry agents. *arXiv preprint arXiv:2510.17491v1*, 2025. URL http://arxiv.org/abs/2510.17491v1.

[99] Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387v2*, 2024. URL http://arxiv.org/abs/2404.14387v2.

[100] Samuel M. Taylor and Benjamin K. Bergen. Do large language models exhibit spontaneous rational deception? *arXiv preprint arXiv:2504.00285v1*, 2025. URL http://arxiv.org/abs/2504.00285v1.

[101] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Automl-agent: A multi-agent llm framework for full-pipeline automl. *arXiv preprint arXiv:2410.02958v2*, 2024. URL http://arxiv.org/abs/2410.02958v2.

[102] 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.

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

[104] Chenyu Wang, Weixin Luo, Sixun Dong, Xiaohua Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. Mllm-tool: A multimodal large language model for tool agent learning. *arXiv preprint arXiv:2401.10727v3*, 2024. URL http://arxiv.org/abs/2401.10727v3.

[105] Lingzhi Wang, Xinyi Shi, Ziyu Li, Yi Jiang, Shiyu Tan, Yuhao Jiang, Junjie Cheng, Wenyuan Chen, Xiangmin Shen, Zhenyuan LI, and Yan Chen. Automated penetration testing with llm agents and classical planning. *arXiv preprint arXiv:2512.11143v1*, 2025. URL http://arxiv.org/abs/2512.11143v1.

[106] Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651v2*, 2024. URL http://arxiv.org/abs/2402.11651v2.

[107] Yun Wang, Zhaojun Ding, Xuansheng Wu, Siyue Sun, Ninghao Liu, and Xiaoming Zhai. Autoscore: Enhancing automated scoring with multi-agent large language models via structured component recognition. *arXiv preprint arXiv:2509.21910v1*, 2025. URL http://arxiv.org/abs/2509.21910v1.

[108] Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Song. Agentvigil: Generic black-box red-teaming for indirect prompt injection against llm agents. *arXiv preprint arXiv:2505.05849v4*, 2025. URL http://arxiv.org/abs/2505.05849v4.

[109] Taylor Webb, Shanka Subhra Mondal, and Ida Momennejad. Improving planning with large language models: A modular agentic architecture. *arXiv preprint arXiv:2310.00194v5*, 2023. URL http://arxiv.org/abs/2310.00194v5.

[110] Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-memguard: A proactive defense framework for llm-based agent memory. *arXiv preprint arXiv:2510.02373v1*, 2025. URL http://arxiv.org/abs/2510.02373v1.

[111] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857v1*, 2025. URL http://arxiv.org/abs/2511.20857v1.

[112] Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538v1*, 2026. URL http://arxiv.org/abs/2601.12538v1.

[113] Lianggui Weng, Dandan Liu, Rong Zhu, Bolin Ding, and Jingren Zhou. Bridgescope: A universal toolkit for bridging large language models and databases. *arXiv preprint arXiv:2508.04031v1*, 2025. URL http://arxiv.org/abs/2508.04031v1.

[114] 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.

[115] 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.

[116] 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.

[117] 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.

[118] Yaozu Wu, Dongyuan Li, Yankai Chen, Renhe Jiang, Henry Peng Zou, Wei-Chieh Huang, Yangning Li, Liancheng Fang, Zhen Wang, and Philip S. Yu. Multi-agent autonomous driving systems with large language models: A survey of recent advances. *arXiv preprint arXiv:2502.16804v2*, 2025. URL http://arxiv.org/abs/2502.16804v2.

[119] Yicong Wu, Ting Chen, Irit Hochberg, Zhoujian Sun, Ruth Edry, Zhengxing Huang, and Mor Peleg. Lessons learned from evaluation of llm based multi-agents in safer therapy recommendation. *arXiv preprint arXiv:2507.10911v1*, 2025. URL http://arxiv.org/abs/2507.10911v1.

[120] 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.

[121] Jingao Xu, Shuoyoucheng Ma, Xin Song, Rong Jiang, Hongkui Tu, and Bin Zhou. Exemplar-guided planing: Enhanced llm agent for kgqa. *arXiv preprint arXiv:2510.15283v1*, 2025. URL http://arxiv.org/abs/2510.15283v1.

[122] Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See Kiong Ng, and Jiashi Feng. Magic: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. *arXiv preprint arXiv:2311.08562v3*, 2023. URL http://arxiv.org/abs/2311.08562v3.

[123] Tianxiang Xu, Zhichao Wen, Xinyu Zhao, Jun Wang, Yan Li, and Chang Liu. L2m-aid: Autonomous cyber-physical defense by fusing semantic reasoning of large language models with multi-agent reinforcement learning (preprint). *arXiv preprint arXiv:2510.07363v2*, 2025. URL http://arxiv.org/abs/2510.07363v2.

[124] Zhenran Xu, Senbao Shi, Baotian Hu, Jindi Yu, Dongfang Li, Min Zhang, and Yuxiang Wu. Towards reasoning in large language models via multi-agent peer review collaboration. *arXiv preprint arXiv:2311.08152v2*, 2023. URL http://arxiv.org/abs/2311.08152v2.

[125] 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.

[126] 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.

[127] 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.

[128] Wanghao Ye, Sihan Chen, Yiting Wang, Shwai He, Bowei Tian, Guoheng Sun, Ziyi Wang, Ziyao Wang, Yexiao He, Zheyu Shen, Meng Liu, Yuning Zhang, Meng Feng, Yang Wang, Siyuan Peng, Yilong Dai, Zhenle Duan, Lang Xiong, Joshua Liu, Hanzhang Qin, and Ang Li. Cognipair: From llm chatbots to conscious ai agents – gnwt-based multi-agent digital twins for social pairing – dating & hiring applications. *arXiv preprint arXiv:2506.03543v2*, 2025. URL http://arxiv.org/abs/2506.03543v2.

[129] 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.

[130] Yue Yin. Infobid: A simulation framework for studying information disclosure in auctions with large language model-based agents. *arXiv preprint arXiv:2503.22726v1*, 2025. URL http://arxiv.org/abs/2503.22726v1.

[131] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *arXiv preprint arXiv:2311.13743v2*, 2023. URL http://arxiv.org/abs/2311.13743v2.

[132] 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.

[133] Rasoul Zahedifar, Sayyed Ali Mirghasemi, Mahdieh Soleymani Baghshah, and Alireza Taheri. Llm-agent-controller: A universal multi-agent large language model system as a control engineer. *arXiv preprint arXiv:2505.19567v1*, 2025. URL http://arxiv.org/abs/2505.19567v1.

[134] 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.

[135] 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.

[136] 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.

[137] 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.

[138] Ke Zhang, Xiaoning Zhao, Ce Zheng, Jiahong Ning, Dandan Zhu, Wenqi Zhang, Chen Sun, and Toshiharu Sugawara. Tool-roco: An agent-as-tool self-organization large language model benchmark in multi-robot cooperation. *arXiv preprint arXiv:2511.21510v2*, 2025. URL http://arxiv.org/abs/2511.21510v2.

[139] Minxing Zhang, Yi Yang, Roy Xie, Bhuwan Dhingra, Shuyan Zhou, and Jian Pei. Generalizability of large language model-based agents: A comprehensive survey. *arXiv preprint arXiv:2509.16330v1*, 2025. URL http://arxiv.org/abs/2509.16330v1.

[140] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618v1*, 2025. URL http://arxiv.org/abs/2510.04618v1.

[141] 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.

[142] Yiqun Zhang, Xiaocui Yang, Xingle Xu, Zeran Gao, Yijie Huang, Shiyi Mu, Shi Feng, Daling Wang, Yifei Zhang, Kaisong Song, and Ge Yu. Affective computing in the era of large language models: A survey from the nlp perspective. *arXiv preprint arXiv:2408.04638v2*, 2024. URL http://arxiv.org/abs/2408.04638v2.

[143] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501v1*, 2024. URL http://arxiv.org/abs/2404.13501v1.

[144] 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.

[145] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. *arXiv preprint arXiv:2305.16986v3*, 2023. URL http://arxiv.org/abs/2305.16986v3.

[146] 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.

[147] 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.

[148] 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.

[149] 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.

[150] Zhilun Zhou, Yuming Lin, and Yong Li. Large language model empowered participatory urban planning. *arXiv preprint arXiv:2402.01698v1*, 2024. URL http://arxiv.org/abs/2402.01698v1.

[151] Jiachen Zhu, Menghui Zhu, Renting Rui, Rong Shan, Congmin Zheng, Bo Chen, Yunjia Xi, Jianghao Lin, Weiwen Liu, Ruiming Tang, Yong Yu, and Weinan Zhang. Evolutionary perspectives on the evaluation of llm-based ai agents: A comprehensive survey. *arXiv preprint arXiv:2506.11102v1*, 2025. URL http://arxiv.org/abs/2506.11102v1.

[152] 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.

[153] Yakun Zhu, Shaohang Wei, Xu Wang, Kui Xue, Xiaofan Zhang, and Shaoting Zhang. Menti: Bridging medical calculator and llm agent with nested tool calling. *arXiv preprint arXiv:2410.13610v3*, 2024. URL http://arxiv.org/abs/2410.13610v3.

[154] Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, Yangning Li, Dongyuan Li, Renhe Jiang, Xue Liu, and Philip S. Yu. Llm-based human-agent collaboration and interaction systems: A survey. *arXiv preprint arXiv:2505.00753v4*, 2025. URL http://arxiv.org/abs/2505.00753v4.