Self-Evolving Context Engineering for LLM-Based Multi-Agent Systems: A Survey

Anonymous ACL submission

Abstract

LLM-based agents are increasingly deployed in tasks that are long-horizon, tool-intensive, multi-role, and subject to safety or auditability constraints. Multi-agent systems can improve specialization and verification, but their reliability depends less on the number of agents alone than on how task context is selected, routed, validated, updated, and reused. This survey studies *self-evolving context engineering*: the governed updating of non-parametric context artifacts such as prompts, memories, playbooks, role profiles, workflow rules, tool descriptions, retrieval policies, and communication topologies. We organize recent work into a three-layer taxonomy covering context construction, multi-agent context coordination, and context evolution. We then compare how these mechanisms appear in software engineering, healthcare, finance, autonomous driving, scientific discovery, creative generation, and safety-critical deployment. Across domains, we identify recurring tensions among capability, provenance, privacy, cost, and rollback. Finally, we summarize evaluation desiderata for measuring context relevance, collaboration gain, memory contamination, update quality, safety incidents, and deployment overhead. The survey argues that robust industrial agents require auditable context lifecycles, not only stronger backbone models.

1 Introduction

Large language models (LLMs) are increasingly used as agents that retrieve information, call tools, interact with environments, maintain state, and revise behavior from feedback (Schick et al., 2023; Yao et al., 2023; Shinn et al., 2023; Madaan et al., 2023; Park et al., 2023; Packer et al., 2023). LLM-based multi-agent systems extend this paradigm by assigning distinct roles, tools, memories, or objectives to different agents. Existing work has explored role-play collaboration, programmable conversations, software-engineering workflows, agent environments, dynamic hierarchies, proposal aggregation, and topology optimization (Li et al., 2023a; Wu et al., 2024; Hong et al., 2024; Qian et al., 2024; Chen et al., 2024; Wang et al., 2025b,a,e).

This shift is especially consequential for industry-facing tasks. Software maintenance, clinical decision support, financial analysis, autonomous driving, scientific discovery, and creative production are rarely single-turn or self-contained. They require domain evidence, tool execution, intermediate artifacts, role-specific expertise, compliance constraints, and traceable decisions. Multi-agent systems can better resemble human organizations than a single monolithic agent, but they also create a new coordination problem: each agent must receive the right information at the right time, trust the right evidence, expose the right intermediate artifacts, and avoid preserving harmful feedback.

We argue that this coordination problem is fundamentally a problem of *context*. In a multi-agent system, context includes instructions, demonstrations, retrieved documents, memories, tool outputs, peer-agent messages, evaluator feedback, domain policies, and human corrections. Research on in-context learning shows that demonstrations, labels, templates, ordering, and prompt composition shape model behavior (Brown et al., 2020; Min et al., 2022; Dong et al., 2024), and rationale-driven collaborative few-shot prompting further highlights that example quality and coordination strategy jointly influence downstream annotation reliability (Wu et al., 2025a). Retrieval-augmented generation treats external documents as part of the model's usable input (Lewis et al., 2020), while long-context and memory benchmarks show that simply adding more tokens does not guarantee reliable evidence use or persistence (Bai et al., 2024;

Figure 1: Taxonomy-style roadmap of self-evolving context engineering, redrawn in the forest layout. Rotated nodes form the survey spine; leaf boxes summarize mechanisms, evaluation hooks, domains, and representative top-venue references.

Liu et al., 2024a; Chen et al., 2025; Tan et al., 2025). Recent context-engineering work therefore broadens the unit of design from a prompt string to a dynamic information substrate that can be selected, compressed, validated, stored, and reused (Mei et al., 2025).

This survey focuses on *self-evolving context engineering*: the feedback-driven updating of non-parametric context artifacts, including prompts, demonstrations, memories, playbooks, role profiles, workflow rules, tool descriptions, retrieval policies, routing policies, and communication topologies. The definition is intentionally bounded. We do not cover unrestricted model-weight self-modification or autonomous rewriting of safety policies. Instead, we study artifacts that can be inspected, versioned, approved, rolled back, and audited. This scope distinguishes our survey from general surveys of LLM agents, LLM-based multi-agent systems, context engineering, and self-evolving agents (Wang et al., 2024b; Guo et al., 2024; Mei et al., 2025; Gao et al., 2026).

Figure 1 provides a process-style illustration of this scope: heterogeneous domain evidence must become typed, role-scoped context, and feedback must pass validation before it becomes persistent memory or a reusable playbook.

We organize the survey around four research questions: (i) what context should enter an industrial agent system, (ii) how context should flow across specialized agents, (iii) which contextual artifacts may evolve after feedback, and (iv) how such updates can be evaluated and governed. Our contributions are:

- We define a three-layer taxonomy of *context construction*, *multi-agent context coordination*, and *context evolution*.
- We explicitly position the survey against adjacent surveys on autonomous agents, multi-agent LLMs, ICL/context engineering, self-evolving agents, domain-specific agents, and agent evaluation.
- We synthesize mechanisms rather than listing papers, emphasizing the artifacts being changed, the feedback signals used, and the safeguards required for reuse.
- We compare domain differences across software engineering, healthcare, finance, autonomous driving, scientific discovery, creative generation, and safety-critical deployment.
- We identify evaluation and governance requirements, including provenance completeness, collaboration gain, memory contamination, update quality, rollback, privacy exposure, cost, and safety incidents.

The resulting perspective treats context as the information substrate of agent behavior, multi-agent collaboration as a mechanism for routing and transforming that substrate, and self-evolution as governed artifact management rather than open-

ended autonomy.

2 Literature Search, Scope, and Selection Methodology

We use a structured narrative review protocol rather than a PRISMA-style systematic meta-analysis. The unit of analysis is a *mechanism*: a reusable design pattern for constructing, routing, or evolving context, not a specific architecture or score. We searched ACL Anthology, OpenReview, arXiv, Google Scholar, Semantic Scholar, and major NLP/ML, systems, HCI, security, and domain venues for work from 2020–2026, retaining earlier foundations such as few-shot learning, RAG, tool use, and agent safety. We prioritized peer-reviewed work for central claims and used recent preprints only to mark frontier mechanisms. Search terms covered LLM agents, multi-agent LLMs, ICL, RAG, long context, memory, self-evolution, agent benchmarks, safety, auditability, and industry domains. Appendix H gives the full coding protocol and paper grouping.

We included work if it constructs, validates, routes, stores, or evolves context; changes information flow among agents; updates non-parametric artifacts such as prompts, memories, playbooks, profiles, workflows, tool descriptions, routing policies, or topologies; provides evaluation logs; or instantiates these mechanisms in industry-like tasks. We excluded incidental wrappers, one-off prompts, unverifiable bibliographic claims, and unrestricted model-weight self-modification. Retained papers were coded by layer, artifact type, update target, feedback signal, validation mechanism, domain, autonomy boundary, and governance constraint. To align with top-tier survey standards, we used an evidence hierarchy (peer-reviewed work for central claims, preprints primarily for frontier mechanism signaling) and a two-pass coding check: uncertain papers were re-coded against the same schema before final placement in the taxonomy and comparison tables.

3 Positioning Against Related Surveys and Frameworks

Existing surveys provide essential foundations, but they optimize along different axes. Autonomous-agent surveys organize the field around agent modules such as planning, memory, tools, and applications (Wang et al., 2024b). LLM-based MAS surveys organize it around roles, communication, debate, hierarchy, and collaboration (Guo et al., 2024). ICL and context-engineering surveys study how information is selected, formatted, retrieved, and reused, usually without making multi-agent routing and persistent update governance the primary object (Dong et al., 2024; Mei et al., 2025). Self-evolving-agent surveys cover a broader adaptation space, including models, tools, environments, prompts, memory, and architectures (Gao et al., 2026). Domain surveys for finance, autonomous driving, and creative MAS provide detailed application taxonomies, but they do not offer a shared cross-domain control plane for context provenance, permissioned routing, update approval, and rollback (Dong et al., 2025; Wu et al., 2025c; Lin et al., 2025). Evaluation surveys and benchmarks catalog agent capabilities, but only partially expose the complete path from context acquisition to inter-agent routing, memory write, approval, reuse, and rollback (Mohammadi et al., 2025; Zhu et al., 2025). Recent systems sharpen the boundary: AutoGen and MetaGPT expose programmable conversation and SOP-based collaboration, MoA studies layered proposal aggregation, AgentDropout optimizes communication topology, and ACE treats context as an evolving playbook (Wu et al., 2024; Hong et al., 2024; Wang et al., 2025a,e; Zhang et al., 2026a). We use recent preprints such as LSS only as frontier framing (Zhang et al., 2026b).

The detailed comparison table is provided in Appendix E to keep the main paper within the ARR page limit.

This survey does not replace these lines of work. It reorganizes them around an industry-oriented multi-agent context lifecycle. The key scope deltas are: (i) we treat context as an operational substrate with explicit types, validation gates, and reuse rules, not as a free-text prompt or a single retrieved passage; (ii) we treat multi-agent coordination as context routing between typed roles with permissioned visibility, rather than as a debate or dialogue protocol; and (iii) we treat self-evolution as governed, non-parametric artifact updating with provenance, approval, and rollback, rather than as open-ended self-modification. The evolvable units are prompts, demonstrations, memories, playbooks, role profiles, workflow rules, tool descriptions, retrieval policies, routing policies, and communication topologies. This focus brings prove-

nance, approval, rollback, cost, safety, and compliance into the technical core rather than leaving them as deployment afterthoughts.

The resulting novelty is therefore a synthesis contribution rather than a claim that the component mechanisms are individually new. We connect four threads that are usually separated: (1) NLP context mechanisms such as ICL, RAG, long-context reading, memory, and factual grounding; (2) MAS mechanisms such as role assignment, message graphs, debate, workflow handoffs, aggregation, hierarchy, and topology adaptation; (3) self-evolution mechanisms such as prompt optimization, playbook curation, profile updates, and topology search; and (4) deployment controls such as provenance, privacy, access control, audit logging, cost budgets, approval, quarantine, and rollback. The framing is also a natural fit for an NLP venue: the artifacts under study are linguistic (prompts, demonstrations, instructions, playbooks, memory entries, judge critiques, role descriptions), the failure modes are language-grounded (conflicting evidence, hallucinated provenance, indirect prompt injection), and the evaluation desiderata extend established NLP threads on in-context learning, RAG, long-context reading, and grounded generation.

4 Taxonomy

We define self-evolving context engineering as the process through which LLM-based multi-agent systems construct, select, share, compress, verify, update, and reuse task-relevant contextual information based on interaction history, tool feedback, peer-agent communication, evaluator judgment, human feedback, and domain-specific outcomes. The three-layer taxonomy is designed to be operational: each layer corresponds to artifacts that can be logged, inspected, or ablated. *Context Construction* asks what information enters the system. *Multi-Agent Context Coordination* asks how information flows across specialized agents. *Context Evolution* asks which reusable artifacts are updated after feedback.

The three layers form a closed operational lifecycle: industry requirements guide context construction, coordinated agents exchange and transform context during task execution, and evaluation signals govern the evolution of reusable contextual artifacts.

The boundaries between the three layers are operational rather than purely conceptual. Construction covers the contextual inputs available before an agent acts, including instructions, demonstrations, retrieved evidence, memory entries, tool outputs, environment observations, structured records, and domain rules. Coordination covers how context is exposed, exchanged, transformed, and constrained during an episode, including role visibility, conversations, workflow artifacts, debate or judge loops, hierarchical routing, graph topology, and permissioned communication. Evolution covers the reusable artifacts that may be updated after feedback, including prompts, examples, memories, playbooks, role profiles, workflow rules, tool descriptions, retrieval policies, routing policies, and topologies. The same object may move across layers. For example, a judge critique is coordination context during the current episode, but it becomes an evolution signal only if it is validated, approved, versioned, and stored for reuse. This distinction prevents a common ambiguity in agent papers: not every runtime message is memory, and not every feedback signal should become future context.

To avoid under-covering the design space, we track four orthogonal dimensions for every mechanism. **Context objects** include instructions, demonstrations, retrieved passages, database rows, tool manuals, API schemas, code slices, simulator states, environment observations, user profiles, domain policies, peer messages, judge critiques, and human corrections. **Context operations** include retrieval, ranking, filtering, normalization, conflict resolution, compression, redaction, routing, summarization, aggregation, critique, memory writing, replay, expiration, and rollback. **Feedback signals** include unit tests, execution traces, human edits, judge scores, peer disagreement, simulator events, delayed clinical or financial outcomes, user ratings, attack logs, privacy scans, and cost or latency budgets. **Governance metadata** includes source, timestamp, owner, visibility scope, consent or license status, freshness window, risk label, validation state, version ID, approval state, and rollback pointer. These dimensions let us cover a wider set of papers without forcing every system into the same architecture.

Compared with agent-, MAS-, or self-evolving-model-centric taxonomies, our unit of analysis is the context lifecycle: what enters the system, how it flows across agents, what changes after feedback, and how those updates remain auditable.

This perspective also exposes cross-layer failure modes: poor construction introduces stale evidence, poor coordination leaks or misroutes context, and poorly governed evolution preserves errors into future episodes.

Appendix C expands the layer-by-layer mechanism map, and Appendix D links evolvable artifacts to update signals, validation gates, risks, and controls.

5 Technical Review

This section reviews mechanisms rather than papers one by one. A single system may use RAG for construction, SOPs for coordination, and tests or judge feedback for evolution, so we emphasize the mechanism, the artifact being changed, and the governance condition for reuse. Across layers, richer context can improve capability but also increases noise, leakage, conflict, cost, and persistent error.

5.1 Context Construction

Context construction has moved from few-shot prompting to heterogeneous information substrates. ICL conditions a frozen LLM on instructions and demonstrations (Brown et al., 2020); subsequent work shows that examples also specify label spaces, formats, distributions, and reasoning styles (Min et al., 2022; Dong et al., 2024). In multi-agent systems, role-specific demonstrations can teach a tester how to report failures, a clinician agent how to ask for missing symptoms, or a finance agent how to separate evidence from recommendation.

RAG extends construction to external evidence such as code, guidelines, filings, regulations, papers, and tool manuals (Lewis et al., 2020); recent black-box and search-augmented variants show how retrieved documents and fresh web evidence can be organized directly into prompts without changing model weights (Shi et al., 2024; Vu et al., 2024). Long-context systems make larger evidence sets possible, but benchmarks show that more tokens do not guarantee reliable evidence use (Bai et al., 2024); excessive or poorly placed context can distract agents, fragment evidence, or hide relevant information (Liu et al., 2024a; Chen et al., 2025; Tao et al., 2025). Memory systems store preferences, failure traces, prior patches, or operating procedures (Tan et al., 2025), while tool traces turn compiler errors, database results, simulator states, clinical tests, and lab measurements into live inputs (You et al., 2025; Boiko et al., 2023). Industry systems should therefore treat context as typed objects with source, timestamp, owner, visibility, validation status, freshness window, and risk metadata. A practical pipeline retrieves or observes candidate context, normalizes it, filters it by policy and freshness, resolves conflicts, compresses it with provenance retained, and binds it to the role that may use it; knowledge-conflict surveys motivate this provenance-heavy treatment even when the concrete conflict taxonomy remains application dependent (Xu et al., 2024b).

Remarks. Construction is not "adding more text". It is selecting, typing, grounding, validating, and bounding evidence before agents act.

5.2 Multi-Agent Context Coordination

Coordination is context routing. Role-based systems give agents different instructions, tools, memories, and local views. CAMEL demonstrates role-play-based collaboration among agents (Li et al., 2023a), while AgentVerse studies multi-agent environments and coordination structures (Chen et al., 2024). AutoGen further supports programmable multi-agent conversation and flexible interaction patterns (Wu et al., 2024). Workflow systems make handoffs explicit through requirements, designs, code, tests, and reviews. MetaGPT organizes collaboration through software-engineering-inspired procedures (Hong et al., 2024), while ChatDev models development as a structured multi-agent workflow (Qian et al., 2024). These artifacts improve auditability because later agents consume structured context rather than an undifferentiated chat history.

Debate, critic, judge, and verifier mechanisms route evaluative context: peer review can expose disagreement, and judge feedback can connect coordination to later evolution, although judge messages should not enter memory without validation. Mixture-of-Agents shows a recent proposal-aggregation design in which outputs from earlier model agents become auxiliary context for later aggregators (Wang et al., 2025a). Graph and hierarchy mechanisms address scale. Scaling studies show that more agents require careful orchestration rather than naive expansion (Qian et al., 2025); resilience analysis shows that faulty agents can destabilize collaboration (Huang et al., 2025);

MegaAgent dynamically creates roles and groups (Wang et al., 2025b); and AgentDropout removes redundant agents and edges to reduce cost and noise (Wang et al., 2025e). MultiAgentBench shows that planning, communication, coordination, topology, and competition require process-level metrics (Zhu et al., 2025). We therefore compare MAS performance to a resource-equivalent single-agent baseline, $\Gamma = \Phi_M/\Phi_S$, to separate synergy from resource accumulation.

Industry coordination also requires permissioned communication. A medical summarizer, trading analyst, tool executor, and compliance reviewer should not share the same read/write permissions. Routing policies should specify which agents may read, write, summarize, forward, or persist each context object. Failure logs should reveal whether the cause was stale retrieval, an unsafe tool result, a weak role prompt, an over-trusted peer message, or an unreliable judge. Production systems should preserve logs that allow independent auditing rather than trusting attribution labels alone.

Remarks. Good coordination increases specialization and verification while limiting redundant communication, unsafe visibility, cost, and latency.

5.3 Context Evolution

Context evolution updates the artifacts that condition future behavior. Reflection and self-refinement revise strategies or outputs through natural-language feedback (Shinn et al., 2023; Madaan et al., 2023). 360°REA, SiriuS, and recent EvolveR-style preprints study multi-perspective assessment, bootstrapped reasoning, and experience lifecycles from accumulated experience (Gao et al., 2024; Zhao et al., 2025; Wu et al., 2025b). Prompt and context optimization methods such as DSPy, PromptBreeder, TextGrad, and ACE treat prompts, modules, or playbooks as editable artifacts optimized by feedback, search, gradients expressed in language, or curation (Khattab et al., 2024; Fernando et al., 2024; Yuksekgonul et al., 2025; Zhang et al., 2026a). Gödel Agent extends this line toward self-referential agent redesign, which is relevant to the survey only under explicit bounded-update and approval assumptions (Yin et al., 2025).

Evolution can also target profiles, workflows, tools, and topologies. MorphAgent-like profile

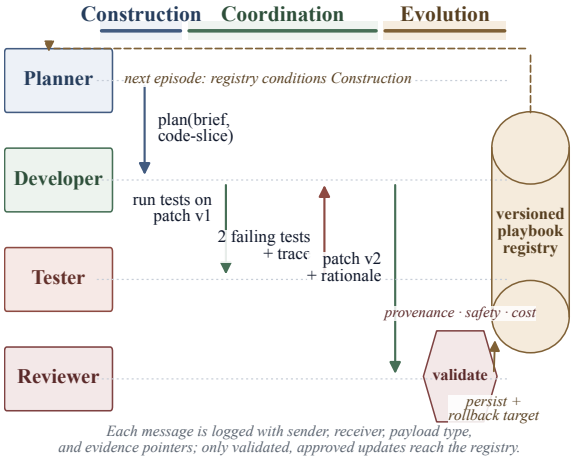

Figure 2: Worked example: a MetaGPT/SWE-agent-style team turns a failed patch into a governed, reusable playbook. Construction retrieves typed issue and repository evidence, coordination routes role-scoped messages across planner, developer, tester, and reviewer agents, and evolution registers a playbook only after validation, approval, and rollback metadata are available.

evolution and LSS-style runtime rewiring remain useful frontier examples but should be treated as recent preprints; stronger venue-backed evidence currently comes from CoMAS for co-evolving MAS and AgentDropout for communication-structure optimization (Lu et al., 2024b; Zhang et al., 2026b; Xue et al., 2026; Wang et al., 2025e). These mechanisms are useful because updated objects are readable and reusable, but they introduce risks: overfitting, reward hacking, benchmark leakage, memory pollution, context collapse, unsafe self-modification, and hidden propagation of poisoned context. We therefore define self-evolution as bounded artifact updating. Every update should record its evidence, target artifact, validation signal, approval status, version ID, and rollback path, and candidate updates should remain separate from accepted updates. Online evolution should be limited to low-risk artifacts; high-stakes policies, clinical rules, trading authority, and safety envelopes require offline validation and human approval.

Remarks. The practical question is not whether an agent can change itself, but which artifacts may change, under what evidence, and how the change can be audited and reversed.

5.4 Worked Example: From a Failed Patch to a Reusable Playbook

Figure 2 instantiates the framework in a software-maintenance schematic in the style of MetaGPT and SWE-agent (Hong et al., 2024; Yang et al., 2024). Construction retrieves typed issue, test, repository, and dependency evidence; coordination routes only the necessary views to planner, developer, tester, and reviewer roles; and evolution stores a debugging playbook only after tests and reviewer approval.

The example illustrates why runtime judge messages are not automatically memory, and why accepted playbooks need provenance, held-out validation, and a rollback target.

6 Industry Applications and Domain Differences

Industry domains differ in feedback speed, evidence reliability, allowed autonomy, and governance requirements. We compare them by context sources, agent roles, feedback signals, evolvable artifacts, and artifacts that must remain gated by policy or human review. Figure 3 gives the schematic feedback-speed/governance view, while Appendix G provides the full cross-domain table. The goal is coverage by mechanism rather than an application catalogue.

Fast-feedback domains. Software engineering and data science are long-horizon but well instrumented: repositories, notebooks, tests, execution logs, and benchmark scores provide concrete feedback. ChatDev and MetaGPT model software development as structured collaboration, while SWE-bench, SWE-agent, and DatawiseAgent expose repository, execution, and notebook provenance (Qian et al., 2024; Hong et al., 2024; Jimenez et al., 2024; Yang et al., 2024; You et al., 2025). Debugging hints, retrieval policies, test prompts, and notebook cleanup rules may evolve; execution privileges, dependency policies, data access, and deployment rules should remain gated.

High-stakes and delayed-feedback domains. Healthcare, finance, autonomous driving, and scientific discovery require slower update gates. AI Hospital and TriageAgent stress active clinical context acquisition; finance-agent systems stress freshness, non-stationarity, compliance-aware memory, and risk disclosure; driving systems stress multimodal sensor context, simulation,

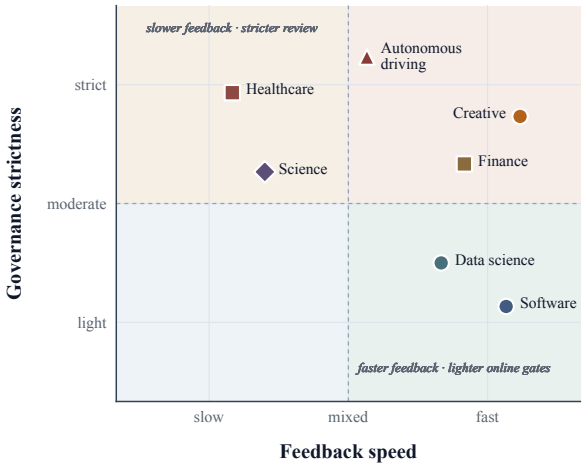

Figure 3: Domain map by feedback speed and governance strictness. The placement is schematic and highlights why self-evolving context needs domain-specific update gates.

and safety envelopes; and science agents combine literature, code, instruments, simulations, protocols, and experimental feedback (Fan et al., 2025; Lu et al., 2024a; Li et al., 2025; Dong et al., 2025; Satapathy et al., 2025; Wu et al., 2025c; Hou et al., 2025; Boiko et al., 2023; Liu et al., 2024b). In these domains, question-asking strategies, scenario memories, retrieval rules, and hypothesis templates may evolve, whereas diagnosis thresholds, medication rules, trading authority, online control policies, hazardous procedures, and safety rules require expert review and provenance-preserving records.

Enterprise, creative, and safety domains. Enterprise agents operate across web pages, GUIs, tickets, databases, and service APIs; WebArena, OSWorld, τ-bench, and ToolSandbox stress realistic control and stateful tool constraints (Zhou et al., 2024; Xie et al., 2024; Yao et al., 2024; Lu et al., 2025). Creative MAS require preference tracking without collapsing diverse styles or reusing private assets (Lin et al., 2025; Ye et al., 2025). Across all domains, OWASP guidance, IPIGuard, MCIP, MCP threat modeling, ToolEmu, Agents Under Siege, adversarial collaboration studies, and AgentDID-style identity mechanisms show that context can carry injection, private data, unsafe instructions, or unauthenticated actions (OWASP Top 10 for Large Language Model Applications Project, 2023; An et al., 2025; Jing et al., 2025; Huang et al., 2026; Ruan et al., 2024; Shahroz et al., 2025; Zhang et al.,

2025; Xu et al., 2026). Persistent updates therefore need quarantine, sandboxing, permissioned memory, and approval.

Remarks. The domain comparison suggests that self-evolution is not a single capability. Software permits fast executable feedback; enterprise agents stress stateful tool and identity constraints; healthcare and driving require conservative validation; finance requires non-stationarity controls; science requires provenance; creative generation requires preference tracking; and safety-critical deployment requires permissioned memory and tool use.

7 Evaluation and Challenges

Evaluation should measure not only final task success, but also whether context was relevant, fresh, attributable, safe, and reusable. Existing benchmarks cover important fragments: general agentic problem solving (Liu et al., 2024c; Mialon et al., 2024), realistic web or GUI control (Zhou et al., 2024; Xie et al., 2024), stateful tool use (Yao et al., 2024; Lu et al., 2025), multi-agent collaboration (Zhu et al., 2025; Xu et al., 2024a), executable software feedback (Jimenez et al., 2024; Yang et al., 2024), and context quality for memory, long context, RAG, and factuality (Tan et al., 2025; Bai et al., 2024; Chen et al., 2025; Es et al., 2024; Saad-Falcon et al., 2024; Min et al., 2023). Yet these resources rarely expose the full chain from context acquisition to inter-agent routing, memory write, approval, reuse, and rollback. A useful protocol should separate seed, update-generation, held-out, shifted, and future reuse tasks; log accepted, rejected, and rolled-back updates; and compare frozen versus evolving context, single-agent versus budget-matched multi-agent runs, memory-write disabled versus enabled, and judge-free versus judge-assisted updates. Appendix I and Appendix J give the operational metrics and full ablation protocol.

Challenge 1: self-evolving context in the wild. Real deployments contain changing tools, users, policies, and data distributions; useful updates may become stale or unsafe, so monitoring and rollback are core requirements.

Challenge 2: domain-aware evolution. Artifacts evolve at different speeds: a debugging hint may follow tests, while clinical, trading, or driving rules require expert review or offline validation.

Challenge 3: downstream evaluation. Industry tasks need executable, auditable outcomes; benchmarks should test acquisition, provenance, delayed feedback, and distribution shift rather than single-round accuracy alone.

Challenge 4: attribution. Failures may originate from stale retrieval, contaminated memory, a weak role prompt, unsafe tool output, biased judge, faulty collaborator, or over-trusted peer message (Huang et al., 2025). Benchmarks should therefore expose per-message provenance graphs, counterfactual context replay, and update logs.

Challenge 5: safety, privacy, and compliance. Self-evolving systems can persist poisoned context, leak sensitive memory, or propagate indirect prompt injection; controls include provenance ledgers, permissioned routing, memory-write gates, sandboxing, and approval.

Challenge 6: cost, scalability, and latency. Long context, many agents, judge calls, and memory retrieval are expensive. Topology optimization, adaptive activation, compression, and caching reduce cost but can remove useful evidence or weaken auditability. For pervasive and edge deployment, these trade-offs become more stringent because memory, power, and communication budgets couple directly with context size and coordination frequency (Wang et al., 2025d,c).

8 Conclusion

This survey framed self-evolving context engineering as a unifying lens for LLM-based multi-agent systems. Reliable agentic systems are shaped not only by model capability or agent count, but also by the lifecycle of the contextual artifacts that condition behavior. The proposed three-layer view clarifies this lifecycle: systems must construct trustworthy context, coordinate its flow across specialized agents, and update reusable artifacts only under explicit evidence, validation, and governance constraints.

Self-evolution should therefore be treated as governed artifact management, not unconstrained autonomy. Progress depends on systems that expose evidence, approval, and rollback traces so agents can be judged by both task success and provenance, privacy, cost, safety, and recoverability.

Limitations

This survey has five concrete limitations that we encourage readers to weigh when using it. First, we present a mechanism-oriented narrative review rather than a PRISMA-style systematic meta-analysis: we do not estimate effect sizes or pool reported numbers across studies, because the surveyed systems differ in tasks, backbones, prompts, retrieval indexes, agent counts, judge designs, and log availability. Second, several frontier mechanisms we discuss come from 2025–2026 preprints whose venue status is not yet settled; we therefore use them to identify mechanisms and gaps rather than as deployment evidence, and we mark them explicitly when their claims are cited. Third, the operational boundaries between construction, coordination, and evolution can be applied in more than one way to the same system, since the same object (e.g., a judge critique) can sit in different layers depending on whether it is reused; we mitigate this by tagging artifacts with their validation gate (Appendix D) but cannot eliminate the residual ambiguity. Fourth, the evaluation metrics we recommend (e.g., collaboration gain Γ, context relevance, memory contamination, update acceptance, rollback safety) are operationally defined but not empirically validated at scale; treating them as a target schema for future benchmarks rather than as settled measurements would be appropriate. Fifth, our domain coverage is uneven by design: software, healthcare, and finance are discussed in more depth because they have richer multi-agent literature, whereas autonomous driving, scientific discovery, and creative generation are covered at a workflow level and would benefit from deeper domain-specific surveys.

Ethical Considerations

Self-evolving context engineering creates new risks because it persists, reuses, and transports information across agents and episodes. Concretely, operators should anticipate at least the following failure modes: (i) memory poisoning, where adversarial or low-quality content enters reusable memory and contaminates later episodes; (ii) indirect prompt injection cascading through retrieved documents, tool outputs, or peer-agent messages; (iii) silent persistence of unsafe playbooks or routing rules that were never explicitly approved; (iv) privacy leakage when role-scoped data is summarised or forwarded across agents without permission checks; and (v) jurisdictional or regulatory drift, where artifacts that were compliant at write time become non-compliant when reused. High-stakes deployments should enforce provenance logging, permissioned routing, memory-write gates, sandboxed tool execution, human approval for high-impact artifacts, and explicit rollback before persistent context is reused. We discourage applying self-evolving context to safety envelopes, clinical decision policies, or financial trading authority without offline validation, human sign-off, and an auditable rollback path.

References

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. 2025. IPIGuard: A novel tool dependency graph-based defense against indirect prompt injection in LLM agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1023–1039.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Mislav Balunovic, Luca Beurer-Kellner, Edoardo Debenedetti, Marc Fischer, Florian Tramèr, and Jie Zhang. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 82895–82920.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Tom B. Brown, Benjamin Mann, Nick Ryder, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Pei Chen, Hongye Jin, Cheng-Che Lee, Rulin Shao, Jingfeng Yang, Mingyu Zhao, Zhaoyu Zhang, Qin Lu, Kaiwen Men, Ning Xie, Huasheng Li, Bing Yin, Han Li, and Lingyun Wang. 2025. LongLeader: A comprehensive leaderboard for large language models in long-context scenarios. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8734–8750.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Yifei Dong, Fengyi Wu, Kunlin Zhang, Yilong Dai, Sanjian Zhang, Wanghao Ye, Sihan Chen, and Zhi-Qi Cheng. 2025. Large language model agents in finance: A survey bridging research, practice, and real-world deployment. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17889–17907.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Zhihao Fan, Lai Wei, Jialong Tang, Wei Chen, Wang Siyuan, Zhongyu Wei, and Fei Huang. 2025. AI hospital: Benchmarking large language models in a multi-agent medical interaction simulator. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10183–10213, Abu Dhabi, UAE. Association for Computational Linguistics.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, and 8 others. 2026. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026-January.

Shen Gao, Hao Li, Zhengliang Shi, Chengrui Huang, Quan Tu, Shuo Shang, Zhiliang Tian, and Minlie Huang. 2024. 360°REA: Towards a reusable experience accumulation with 360° assessment for multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13149–13162.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8048–8057.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*.

Xinmeng Hou, Wuqi Wang, Long Yang, Hao Lin, Jinglun Feng, Haigen Min, and Xiangmo Zhao. 2025. Driveagent: Multi-agent structured reasoning with llm and multimodal sensor fusion for autonomous driving. *IEEE Robotics and Automation Letters*, 10(11):12189–12196.

Charoes Huang, Xin Huang, Ngoc Phu Tran, and Amin Milani Fard. 2026. Model context protocol threat modeling and analysis of vulnerabilities to prompt injection with tool poisoning. *Journal of Cybersecurity and Privacy*, 6(3):84.

Jen-Tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael Lyu, and Maarten Sap. 2025. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 26202–26226. PMLR.

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

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues?

Huihao Jing, Haoran Li, Wenbin Hu, Qi Hu, Xu Heli, Tianshu Chu, Peizhao Hu, and Yangqiu Song. 2025. MCIP: Protecting MCP safety via model contextual integrity protocol. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1177–1194.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language

model calls into self-improving pipelines. In *International Conference on Learning Representations*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for mind exploration of large language model society. In *Advances in Neural Information Processing Systems*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. 2025. QuantAgents: Towards multi-agent financial system via simulated trading. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17438–17464.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. 2025. Creativity in LLM-based multi-agent systems: A survey. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27584–27607.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Sizhe Liu, Yizhou Lu, Siyu Chen, Xiyang Hu, Jieyu Zhao, Tianfan Fu, and Yue Zhao. 2024b. DrugAgent: Automating ai-aided drug discovery programming through llm multi-agent collaboration. *arXiv preprint arXiv:2411.15692*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024c. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, Albuquerque, New Mexico. Association for Computational Linguistics.

Meng Lu, Brandon Ho, Dennis Ren, and Xuan Wang. 2024a. TriageAgent: Towards better multi-agents collaborations for large language model-based clinical triage. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5747–5764.

Siyuan Lu, Jiaqi Shao, Bing Luo, and Tao Lin. 2024b. MorphAgent: Empowering agents through self-evolving profiles and decentralized collaboration. *arXiv preprint arXiv:2410.15048*.

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

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

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: A benchmark for general ai assistants. In *International Conference on Learning Representations*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 6129–6139. Association for Computing Machinery.

OWASP Top 10 for Large Language Model Applications Project. 2023. Owasp top 10 for large language model applications. Version 1.1.0 archived PDF and project page maintained by OWASP GenAI Security Project.

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

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565.

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

Chen Qian, Zhenheng Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*.

Yangjun Ruan, Jingwen Dong, Andrew Wang, Zhengxuan Lu, Yu Gu, Tongzhou Wang, Yang Cao, Weiyan Shi, and Zhiting Hu. 2024. ToolEmu: Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Ranjan Satapathy, Raphael Liew, Joyjit Chattorj, Erik Cambria, and Rick Goh. 2025. From earnings calls to investment reports: Evaluating role-based multi-agent LLM systems. In *Proceedings of The 10th Workshop on Financial Technology and Natural Language Processing*, pages 258–267.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551.

Rana Shahroz, Zhen Tan, Sukwon Yun, Charles Fleming, and Tianlong Chen. 2025. Agents under siege: Breaking pragmatic multi-agent LLM systems with optimized prompt attacks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9661–9674.

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

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19336–19352.

Wenyu Tao, Xiaofen Xing, Zeliang Li, and Xiangmin Xu. 2025. SAKI-RAG: Mitigating context fragmentation in long-document RAG via sentence-level attention knowledge integration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1195–1213.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2024. FreshLLMs: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13697–13720, Bangkok, Thailand. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025a. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18:186345.

Qian Wang, Tianyu Wang, Zhenheng Tang, Qinbin Li, Nuo Chen, Jingsheng Liang, and Bingsheng He. 2025b. Megaagent: A large-scale autonomous llm-based multi-agent system without predefined sops. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4998–5036. Association for Computational Linguistics.

Xubin Wang, Qing Li, and Weijia Jia. 2025c. Cognitive edge computing: A comprehensive survey on optimizing large models and ai agents for pervasive deployment. *arXiv preprint arXiv:2501.03265*.

Xubin Wang, Zhiqing Tang, Jianxiong Guo, Tianhui Meng, Chenhao Wang, Tian Wang, and Weijia Jia. 2025d. Empowering edge intelligence: A comprehensive survey on on-device ai models. *ACM Computing Surveys*, 57(9):1–39.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025e. Agent-Dropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24013–24035.

Jianfei Wu, Xubin Wang, and Weijia Jia. 2025a. Enhancing text annotation through rationale-driven collaborative few-shot prompting. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025b. EvolveR: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*.

Yaozu Wu, Dongyuan Li, Yankai Chen, Renhe Jiang, Henry Peng Zou, Wei-Chieh Huang, Yangning Li, Liancheng Fang, Zhen Wang, and Philip S. Yu. 2025c. Multi-agent autonomous driving systems with large language models: A survey of recent advances, resources, and future directions. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12756–12773.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pages 52040–52094.

Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See-Kiong Ng, and Jiashi Feng. 2024a. MAgIC: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7315–7332, Miami, Florida, USA. Association for Computational Linguistics.

Minghui Xu, Xiaoyu Liu, Yihao Guo, Chunchi Liu, Yue Zhang, and Xiuzhen Cheng. 2026. Agent-DID: Trustless identity authentication for ai agents. *Preprint*, arXiv:2604.25189. Accepted by ICDCS 2026.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024b. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. 2026. CoMAS: Co-evolving multi-agent systems via interaction rewards. In *International Conference on Learning Representations*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. Tau-bench: A benchmark for tool-agent-user interaction in real-world domains.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Wen Ye, Zhaocheng Liu, Gui Yuwei, Tingyu Yuan, Yunyue Su, Bowen Fang, Chaoyang Zhao, Qiang Liu, and Liang Wang. 2025. GenPilot: A multi-agent system for test-time prompt optimization in image generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 929–958.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. 2025. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27890–27913.

Ziming You, Yumiao Zhang, Dexuan Xu, Yiwei Lou, Yandong Yan, Wei Wang, Huamin Zhang, and Yu Huang. 2025. DatawiseAgent: A notebook-centric LLM agent framework for adaptive and robust data science automation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1099–1123.

Song Yu, Xiaofei Xu, Ke Deng, Li Li, and Lin Tian. 2025. Tree of agents: Improving long-context capabilities of large language models through multi-perspective reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4574–4592.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616.

Jiahao Zhang, Baoshuo Kan, Tao Gong, Fu Lee Wang, and Tianyong Hao. 2025. When allies turn foes: Exploring group characteristics of LLM-based multi-agent collaborative systems under adversarial attacks. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6275–6300.

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. 2026a. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*.

Weihao Zhang, Yitong Zhou, Huanyu Qu, and Hongyi Li. 2026b. Loosely-structured software: Engineering context, structure, and evolution entropy in runtime-rewired multi-agent systems. *arXiv preprint arXiv:2603.15690*. Recent preprint.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. 2025. SiriuS: Self-improving multi-agent systems via bootstrapped reasoning. In *Advances in Neural Information Processing Systems*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Ou, Yonatan Bisk, Daniel Fried, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 2025. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622.

A Formal Definitions

This section defines the minimal notation needed to read the tables and lifecycle claims.

For implementation-oriented readers, a context item can be represented as $c = (x, \tau, s, t, v, r)$, where x is content, τ is type, s is source, t is timestamp, v is visibility scope, and r is risk/provenance metadata. A multi-agent context flow can be represented as a directed graph $G_t = (A, C, E_t)$ over agents A, context items C, and exposure/update edges E_t. An evolution operator U updates bounded artifacts such as prompts, memories, role profiles, workflows, tool descriptions, or routing policies. In high-stakes domains, U should be constrained by a governance policy Π that specifies permissions, approval, logging, and rollback.

B Timeline of Representative Milestones

Figure 4 provides a compact cross-year map for construction, coordination, evolution, and governance milestones.

C Full Taxonomy Table

Table 1 is the full mechanism lookup table: each row maps a mechanism family to artifact type, feedback signal, and governance implication.

D Artifact Governance Table

Table 2 is the executable gate policy: it specifies which artifacts may evolve online and which require stricter approval, rollback, and logging.

E Related-Work Positioning Table

Table 3 provides a side-by-side scope comparison to support novelty assessment along lifecycle coverage, artifact focus, and governance depth.

F Full Benchmark Table

Table 4 is a coverage map of benchmark evidence by interaction type and governance stressor, and should be read jointly with Table 6.

G Detailed Domain Analysis

This section expands Section 6 with domain-specific operational ingredients and gate strictness.

G.1 Cross-Domain Application Profiles

Figure 5 provides a compact operational map of how context evolution should be governed across representative application settings.

The figure can be interpreted with three governance regimes:

- **Panel A (fast-feedback domains):** software engineering and data science settings can iterate low-risk artifacts (e.g., retrieval rules, workflow hints, prompt variants) more quickly because execution traces and test signals are available.
- **Panel B (high-stakes or delayed-feedback domains):** healthcare, finance, autonomous driving, and science/drug workflows require stricter validation, expert review, and rollback readiness before persistent updates are accepted.
- **Panel C (enterprise and creative applications):** governance must also account for stateful services, identity constraints, private assets, subjective preference feedback, and stronger security controls.

G.2 Cross-Domain Comparison Table

The following table compares context sources, feedback signals, and gate designs across domains.

Table 5 summarizes three practical boundaries:

- Rapid-feedback domains (e.g., software) can iterate more artifacts online.
- Delayed- or safety-critical domains (e.g., healthcare, finance, driving, science) require stronger staging, approval, and rollback gates.
- Identity- and compliance-heavy settings should keep authority-related artifacts under strict governance.

H Search Methodology and Related-Work Positioning

This section states how papers were coded and mapped to claims, sections, and tables.

For each retained paper or framework, we coded:

- Technical layer: construction, coordination, or evolution.
- Artifact type: prompt, retrieval policy, memory, playbook, profile, workflow, tool description, routing policy, topology, benchmark, or safety control.

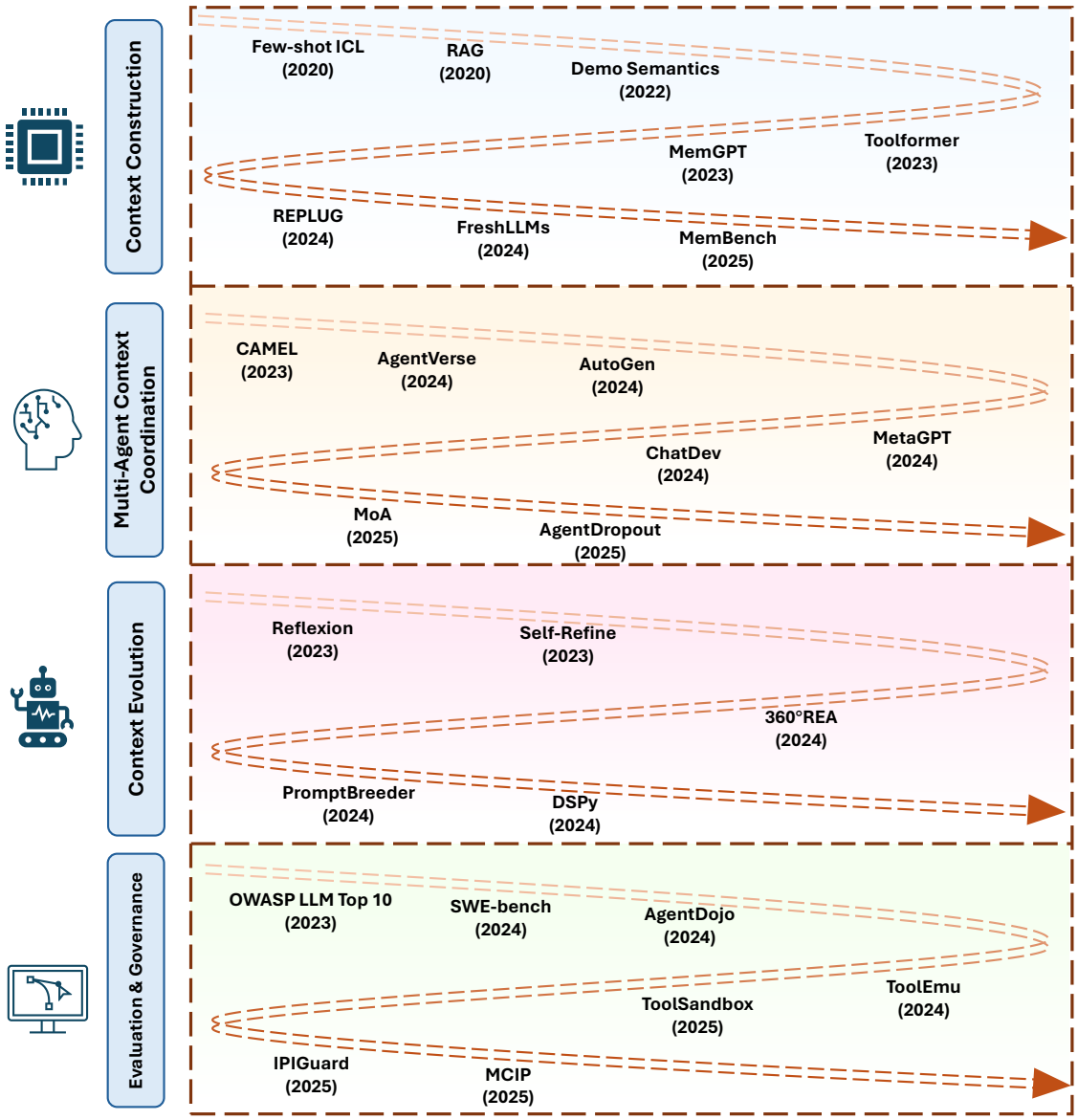

Figure 4: Timeline of representative milestones related to self-evolving context engineering, spanning context construction, multi-agent context coordination, context evolution, and evaluation/governance threads.

- Feedback signal: tool execution, judge, peer critique, human approval, environment state, or delayed domain outcome.
- Domain and governance requirement.

We used these codes to decide whether a paper supports a main-text mechanism, a domain example, an evaluation metric, or appendix-only background. Frontier preprints were kept when they directly address reviewer concerns, but core claims rely primarily on established mechanisms and benchmarks.

To avoid redundancy with the bibliography and the related-work positioning matrix, this appendix reports coding rules and mapping criteria rather than a separate full retained-paper inventory table.

Table 3 then operationalizes these codes into a concise related-work comparison.

I Operational Metrics

This section defines implementation-grade metrics and required logs. Table 6 is a quick checklist, and Table 7 is the full specification aligned with lifecycle stages.

J Ablation Protocol

Table 8 specifies controlled ablations that attribute gains and failures to construction, coordination, or

Layer	Subcategory	Examples	Main risks
Construction	ICL / demonstrations	task examples, role examples, templates	order sensitivity, template bias
Construction	RAG / long context	documents, code, filings, guidelines, papers	noise, conflict, staleness, fragmentation
Construction	Memory / tools	experience records, execution logs, tool outputs	pollution, unsafe tool outputs, privacy leakage
Coordination	Roles / workflows	specialist prompts, SOP handoffs, artifact chains	brittle roles, cascading hallucination
Coordination	Debate / judge	critic messages, consensus, evaluator feedback	judge bias, reward hacking, cost
Coordination	Graph / hierarchy	routers, topologies, dynamic recruitment	missing context, uncontrolled propagation
Evolution	Experience / prompt	playbooks, prompts, examples, memories	overfitting, context collapse
Evolution	Profiles / policies	role profiles, tool descriptions, workflow rules	unsafe self-modification, rollback gaps

Table 1: Expanded taxonomy used for appendix-level mapping.

Artifact class	Examples	Update signals	Validation gate	Main risks	Suggested controls
Instructions / prompts	system prompts, role prompts, task templates	task scores, judge feedback, human edits	held-out validation, format/schema checks, reviewer sign-off	overfitting, hidden policy drift	version IDs, prompt diffs, approval before persistence
Demonstrations	few-shot examples, solved traces, code patches	retrieval success, execution traces, user correction	deduplication, leakage checks, source verification	benchmark leakage, stale examples	split-aware storage, provenance tags, expiration windows
Memory / playbooks	preferences, failure notes, domain heuristics	repeated success/failure, reflection, tool outcomes	confidence thresholds, contradiction review, privacy scan	memory contamination, private data retention	write gates, quarantine, rollback, access scopes
Retrieval policy	query templates, ranker settings, freshness rules	retrieval relevance, evidence support, citation coverage	retrieval audit, freshness checks, conflict checks	stale or irrelevant evidence	source whitelists, timestamp checks, evidence graph logging
Role / workflow	agent profiles, SOPs, handoff schemas	role ablation, communication utility, task latency	simulation checks, dependency checks, owner approval	redundant roles, brittle handoffs	role registry, interface contracts, ablation logs
Tools / permissions	tool descriptions, API scopes, sandbox policy	tool failures, security incidents, compliance review	sandbox tests, least-privilege review, red-team checks	unsafe execution, prompt injection	capability scopes, tool quarantine, call logging
Routing / topology	message graph, aggregation rule, debate path	edge utility, token cost, faulty-agent robustness	budget-matched comparison, topology ablation	cost blow-up, harmful message propagation	permissioned routing, edge budgets, isolation of untrusted agents

Table 2: Common evolvable context artifacts and governance requirements. Self-evolution is constrained to inspectable artifacts that pass explicit validation gates, rather than silently altering deployment behavior.

evolution components under the same logging assumptions as Table 7.

Work / line of work	Main lens	Scope	Evolution target	MAS focus	Industry focus	Governance / auditability	Difference from this survey
Autonomous-agent surveys (Wang et al., 2024b)	Agent modules	Planning, tools, memory, reflection, applications	Broad or secondary	Partial	Mixed	Usually high-level	We focus on typed context artifacts inside industry MAS rather than the full agent stack.
LLM-based MAS surveys (Guo et al., 2024)	Collaboration	Roles, communication, hierarchy, debate, topology	Often fixed prompts or roles	Central	Limited to examples	Limited	We reinterpret MAS as context-routing, exposure control, and verification systems.
ICL / context-engineering surveys (Dong et al., 2024; Mei et al., 2025)	Information design	Examples, retrieval, memory, tools, long context, context optimization	Retrieval, memory, or compression policy	Low to partial	Domain dependent	Partial provenance	We integrate these components into one auditable MAS lifecycle with persistent update gates.
Self-evolving-agent surveys (Gao et al., 2026)	Adaptation	Model, prompt, tool, memory, environment, architecture	Many targets, sometimes model-level	Included	Broad	Broad safety	We restrict evolution to inspectable non-parametric artifacts with approval and rollback.
Domain surveys (Dong et al., 2025; Wu et al., 2025c; Lin et al., 2025)	Application domain	Finance, driving, creative MAS, resources, deployment constraints	Domain-specific artifacts	Varies	Strong within domain	Varies	We provide a cross-domain schema for context sources, feedback speed, autonomy boundaries, and governance gates.
Evaluation surveys / benchmarks (Mohammadi et al., 2025; Zhu et al., 2025)	Measurement	Agent capability, collaboration, reliability, cost, safety	Evaluation target rather than mechanism	Partial	Strong	Strong	We require logs for acquisition, routing, write decisions, reuse, ablation, and rollback.
ACE / LSS-style frameworks (Zhang et al., 2026a,b)	Playbooks and artifacts	Evolving contexts, runtime views, structure, evolution entropy	Playbooks, views, artifacts	Partial to central	Software-leaning	Stronger artifact view	We place these ideas in a wider MAS taxonomy, domain comparison, and evaluation protocol.

Table 3: Positioning against related surveys and frameworks. The table emphasizes differences in lifecycle coverage, evolvable artifact scope, and audit/governance commitments for industry-oriented LLM-based MAS.

Benchmark	Main target	Context/evolution relevance	Limitation for self-evolving context
AgentBench / GAIA	General agents	Tool use and heterogeneous task solving	Limited persistent evolution
WebArena / OSWorld	Web and GUI agents	Environment state and action feedback	High cost; domain shift remains hard
SWE-bench / SWE-agent	Software repair	Executable tests and repository context	May reward benchmark-specific memory
τ-bench / ToolSandbox	Tool/service use	Stateful tool interactions and policy following	Limited multi-agent evaluation
MultiAgentBench / MAgIC	Multi-agent collaboration	Collaboration quality, competition, cognition	Still weak on long-term memory evolution
MemBench / LongBench / LongLeader	Memory and long context	Retention, retrieval, long-document use	Does not fully test governed context updates

Table 4: Benchmarks relevant to evaluating self-evolving context engineering.

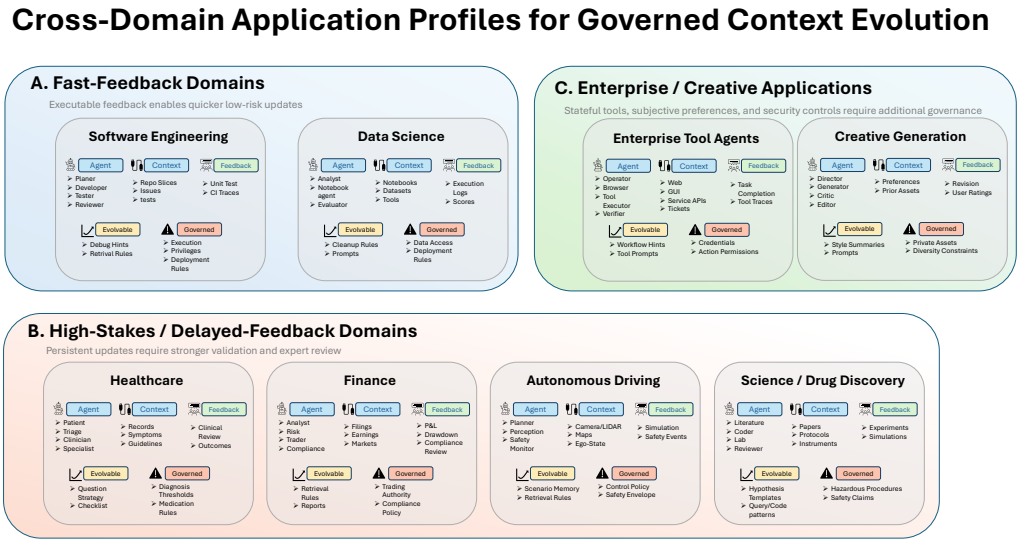

Figure 5: Cross-domain application profiles for governed context evolution. Panel A highlights fast-feedback domains, where executable feedback supports faster low-risk updates. Panel B highlights high-stakes or delayed-feedback domains, where persistent updates require stronger validation and expert review. Panel C highlights enterprise and creative applications, where stateful tools, preference diversity, and security constraints require additional governance controls.

Domain	Task / agents	Context sources	Feedback signals	Evolvable artifacts	Strictly governed
Software / data	planner, developer, tester, reviewer, analyst	repo slices, issues, tests, notebooks, APIs	unit/integration tests, CI status, execution logs	debugging heuristics, retrieval/test prompts, notebook procedures	execution privileges, dependency locks, deployment policies
Healthcare	patient, doctor, specialist, triage, counselor	symptoms, EHR, labs, images, guidelines	clinician review, delayed outcomes	checklists, question strategies, de-identified playbooks	diagnosis policy, medication rules, patient memory
Finance	analyst, risk, trader, writer, compliance	filings, calls, news, prices, risk limits	P&L, drawdown, compliance review	freshness rules, report templates, scenario notes	trading authority, compliance policy, persistent risk rules
Driving / embodied	perception, scene reasoning, planner, safety monitor	cameras, LiDAR, GPS, maps, ego-state	simulation outcomes, latency, safety-critical events	offline scenarios, critique rules, failure taxonomies	online control policies, safety envelope, sensor-trust assumptions
Enterprise / web	browser, tool, support, compliance, supervisor	UI state, tickets, accounts, policies, APIs	service success, user edits, tool errors	form heuristics, escalation playbooks, tool-selection rules	account actions, irreversible transactions, identity assumptions
Science / drug	literature, coder, planner, lab/tool agents	papers, DBs, code, protocols, assays	simulation, experiments, assays	hypothesis templates, protocol memory, query/code patterns	hazardous procedures, claims, experimental protocols
Creative	ideator, critic, prompt optimizer, editor	prompts, images, preferences, examples	user ratings, critiques, A/B tests	prompt variants, style constraints, preference summaries	private assets, consent, diversity of preferences
Security / trust	red-team, guard, verifier, identity agent	retrieved/tool context, policies, credentials, logs	attack/guard logs, audits, incidents	guard rules, quarantine policies, risk labels	access control, memory writes, authentication, audit trails

Table 5: Cross-domain comparison of context sources, feedback channels, and artifact governance. Rows highlight which updates can be evolved under evidence and which controls must remain under strict policy gates.

Metric group	Metrics	Operational definition	Required logs / artifacts	Why it matters
Task	task success; execution pass rate	Solved-task fraction under a task-specific scorer; fraction of actions/tool calls that execute without runtime or policy failure	outputs, gold labels, environment states, tool/test logs	Captures end-task utility and executable reliability.
Collaboration	collaboration gain; role contribution	$\Gamma = \Phi_M / \Phi_S$ against a budget-matched single-agent baseline; performance drop under role/edge/message ablation	MAS score, baseline score, budget logs, role outputs, ablation runs	Distinguishes true coordination gain from resource accumulation and exposes redundant roles.
Coordination cost	communication cost; topology efficiency	Messages, rounds, tokens, and tool calls per solved task; marginal utility per active edge/agent	message graph, token logs, topology traces, latency traces	Prevents bloated communication and over-connected topologies.
Context quality	relevance; freshness; conflict rate	Share of exposed context needed for final decisions; share within validity window; contradictions per evidence set	retrieved chunks, timestamps, labels, evidence graph	Reduces stale, irrelevant, and contradictory context usage.
Provenance	provenance completeness; audit completeness	Share of claims/actions/updates linked to verifiable sources; share of episodes with complete end-to-end trace	provenance ledger, citations, access logs, artifact registry	Supports accountability, reproducibility, and rollback.
Compression / memory	compression loss; memory contamination rate	Critical evidence lost after summarization; persistent entries later judged false, unsafe, stale, private, or benchmark-specific	original/compressed context, memory store, review labels	Protects long-horizon reuse from hidden information loss and memory pollution.
Evolution	update acceptance; rollback frequency; transfer under shift	Candidate updates accepted under gate checks; accepted updates later reverted; shifted/future performance after updates	approvals, version IDs, rollback logs, held-out/shifted task sets	Evaluates whether self-evolution is stable, controllable, and generalizable.
Deployment	latency; token cost; privacy exposure; safety incident rate	Wall-clock and token budget per task/update; sensitive exposures; unsafe actions or policy violations	runtime logs, token logs, guard logs, incident records	Makes deployment trade-offs and governance risk measurable.

Table 6: Compact operational metrics aligned with construction, coordination, and evolution stages. Appendix I provides full per-metric definitions and required logging fields.

Metric group	Metric	Operational definition	Required logs / artifacts	Why it matters
Task	Task success	Fraction of tasks solved under a task-specific scorer or environment success state	Final outputs, gold labels, environment states	Measures downstream utility.
Task	Execution pass rate	Fraction of code, tool calls, or action sequences that execute without runtime or policy failure	Tool logs, tests, runtime traces	Captures executable reliability.
Collaboration	Collaboration gain	$\Gamma = \Phi_M/\Phi_S$ where Φ_M is MAS utility and Φ_S is a resource-equivalent single-agent baseline	MAS score, baseline score, budget logs	Separates synergy from resource scaling.
Collaboration	Communication cost	Number of messages, rounds, inter-agent tokens, and tool calls per successful task	Message logs, token logs, tool traces	Controls bloated coordination.
Collaboration	Role contribution	Performance change when a role, message stream, or edge is removed	Ablation runs, role outputs, topology logs	Identifies redundant or harmful roles.
Collaboration	Topology efficiency	Utility per active edge/agent or marginal gain from adding/removing a communication path	Topology graph, scores, token budgets	Supports cost-aware routing.
Context	Context relevance	Share of exposed context judged necessary for the final decision or action	Retrieved chunks, judge/human labels, final answer links	Reduces distractors and prompt bloat.
Context	Context freshness	Share of context within its validity window or not superseded by newer evidence	Timestamps, version IDs, source freshness metadata	Avoids stale evidence.
Context	Provenance completeness	Share of claims, actions, or updates linked to a source, tool result, memory, peer message, or human input	Citations, provenance ledger, tool/memory IDs	Enables auditability.
Context	Conflict rate	Number/share of contradictory context items per task or evidence set	Evidence graph, contradiction labels, source metadata	Exposes unreliable or inconsistent context.
Context	Compression loss	Information needed for a correct decision that is absent after summarization or compression	Original context, compressed context, answer evidence labels	Tests whether compression hides critical evidence.
Evolution	Memory contamination rate	Persistent entries later judged false, unsafe, stale, private, duplicated, or benchmark-specific	Memory store, review labels, source history	Prevents harmful reuse.
Evolution	Update acceptance rate	Fraction of proposed artifact updates that pass validation and approval gates	Update candidates, validation results, approval records	Measures quality of evolution signals.
Evolution	Rollback frequency	Fraction of accepted updates later reverted or quarantined	Artifact registry, rollback logs, incident reports	Measures stability and governance burden.
Evolution	Transfer under shift	Performance change on held-out, shifted, or future tasks after accepted updates	Seed/dev/held-out/shifted/future splits	Tests generalization beyond memorized tasks.
Deployment	Latency	Wall-clock time per task, per round, or per accepted update	Runtime traces, queue logs	Determines operational feasibility.
Deployment	Token cost	Prompt and completion tokens per task, successful task, or update	Token accounting logs	Tracks scalability and budget.
Deployment	Privacy exposure	Number/share of sensitive fields exposed to unauthorized agents, tools, or logs	Access logs, PII labels, visibility policy	Supports compliance.
Deployment	Safety incident rate	Unsafe tool calls, policy violations, injection successes, or harmful output events per task	Guard logs, sandbox logs, incident records	Quantifies risk.
Deployment	Audit completeness	Fraction of tasks/updates with complete trace from input context to action, feedback, and versioned update	Provenance ledger, artifact registry, approvals	Makes accountability and rollback possible.

Table 7: Full operational metrics for governed self-evolving context.

Ablation	Keep fixed	Diagnostic question	Required artifacts	Governance / cost–safety signal	Expected failure exposed
Frozen vs. evolving context	model, tools, tasks, budget	Do accepted updates improve future tasks beyond static prompts?	artifact versions, seed/dev/held-out splits	Δaccuracy on held-out, Δ\$/task, rollback rate per 100 updates	overfitting to seed, no real transfer
Single agent vs. MAS	token/tool budget and model family	Is collaboration beneficial after controlling for resources?	single-agent baseline, MAS logs, cost logs	accuracy per \$, messages per task, conflicting handoffs	gains explained by extra budget, not coordination
Memory-write disabled	retrieval and execution policy	Does persistent memory help or contaminate later decisions?	memory store, write attempts, review labels	Δaccuracy on future tasks, % writes flagged by validators	memory pollution, false persistence
Judge-free vs. judge-assisted updates	candidate updates and tasks	Do judges improve update quality or introduce bias?	judge rationales, accepted/rejected updates	judge precision/recall on harmful updates, cost per accepted update	judge over-trust, noisy validation
No provenance filtering	retrieval candidates and ranker	Does source/risk metadata prevent harmful context use?	source tags, access logs, final evidence links	% unsafe context surfaced, privacy/licence violations per run	stale, private, or poisoned context
Fixed vs. optimised topology	agents and task set	Are topology changes useful beyond prompt changes?	message graph, edge utility, token logs	messages per task, redundant-edge ratio, Δaccuracy	redundant or harmful communication
Compression disabled vs. enabled	evidence pool and context window	Does compression preserve decision-critical evidence?	original chunks, compressed context, support labels	tokens per task, evidence-recall@k, Δaccuracy	lost evidence, hallucinated summaries
Rollback disabled vs. enabled	accepted updates and incidents	Can harmful updates be contained after discovery?	incident log, version registry, rollback trace	incident containment time, persistence of harmful updates, audit completeness	persistent unsafe behaviour

Table 8: Ablation protocol for evaluating governed self-evolving context. Each row isolates one part of the context lifecycle and reports both a task-level outcome and a governance / cost–safety signal, so that an improvement is not credited unless it survives joint utility, cost, and safety accounting.