Closing the Agent Loop

Governance Starts at Capture: An Epistemic Memory Architecture for Governed Agentic Operations

Megan Anderson

AI ARMY, INC. · aiarmy.co

June 2026

Abstract

Agent memory systems have largely converged on a storage-and-retrieval model: extract salient claims, store them, embed them, and retrieve them by similarity. That model is useful, but it is insufficient for shared human-agent work, where durable state is read from and written to by multiple humans, agents, tools, and model sessions at machine speed. In this regime, memory is not passive storage. It is operational authority: it tells future agents what to believe, what to ignore, what to continue, what to reopen, and what to do next. If ungoverned claims enter memory, they later return as context with the texture of truth. This paper presents a systems design and bounded-evidence account of an epistemically annotated memory architecture for governed human-agent operations. The design emerged from operator-led R&D across frontier chat models, coding agents, memory/vault workflows, shared task systems, and AI ARMY product development. We do not publish private transcripts as data; instead, we use sanitized operational incidents as failure-mode archetypes and connect them to broader literature on sycophancy, long-context degradation, persistent memory poisoning, provenance, temporal memory, and agent-tool security. The architecture treats epistemics as properties of the memory object itself. Durable claims carry status, promotion state, confidence, stakes, typed provenance, bitemporal validity, supersession lineage, boundary classification, branch status, and reopen conditions. Writes are append-first and proposal-first: agents may propose memory updates, but promotion into durable memory is a governed, observable act. Retrieval is packet-based: agents receive scoped, bounded, purpose-bound Context Packets rather than open access to the store. The paper describes the failure modes that constrained the design, the four-layer separation it settled into (Constellation, Atlas, Luna, Core OS), the costs and limitations it imposes, and the open problems in moving governed memory from human cadence to machine cadence.

Keywords: agent memory; AI governance; context packets; provenance; bitemporal memory; agentic systems; memory poisoning; human-agent collaboration; governed agentic infrastructure

1. Introduction: Memory Is Operational Authority

The modern agent stack has advanced faster than its accountability layer. Frontier models synthesize, plan, write, debug, search, call tools, and maintain partial continuity across sessions. Connectors and protocols let agents reach files, repositories, tools, CRMs, databases, and other operational systems. Memory systems promise persistence across time. Yet in practical multi-agent work, continuity and trust still depend on a human operator. The operator

remembers which decisions are settled and which were only explored. The operator knows which document superseded the prior plan. The operator notices when a model turns a hypothesis into a claim, catches stale context, sees when a decision was true only under an earlier set of assumptions, and carries working state between models that do not share a governed substrate. The user is still the governance layer.

That arrangement is fragile because the properties that make memory useful also make ungoverned memory dangerous. Memory is durable. It is retrieved automatically. It is treated as context rather than as an uncertain claim. It moves from one session into the next with a presumption of relevance. In a shared human-agent system, remembered content can become a premise for action long after the circumstances that made it true have changed.

This paper argues that agentic memory cannot be designed as storage plus retrieval alone. In shared work, memory is operational authority. It tells future agents what to believe, what to continue, what to ignore, what to reopen, what tools to consider, and which decisions are still binding. That authority requires governance at capture, not only runtime guardrails after the model has reasoned from ungoverned premises.

The design presented here emerged from a sustained operator-led R&D environment: frontier chat models, coding agents, vault workflows, shared task systems, and AI ARMY product development running across multiple ventures. The failures did not prove the architecture. They constrained it. They showed what the architecture had to make impossible, auditable, or at minimum visible before agentic memory could be trusted as a substrate for work.

1.1 Contributions

- A design argument that shared agent memory is operational authority, not passive storage or personalization state.
- A failure-mode taxonomy connecting field observations to broader research on sycophancy, long-context degradation, persistent memory poisoning, stale authority, and tool/security risk.
- An epistemically annotated memory-object model: status, promotion state, confidence, stakes, typed provenance, valid time, transaction time, supersession, boundary class, branch status, and reopen conditions.
- A write-path discipline separating proposal authority from promotion authority, with append-first/provenance-first updates and advisory-first enforcement.
- A retrieval-path discipline in which agents receive scoped Context Packets rather than open access to the memory store.
- A four-layer reference architecture: Constellation preserves knowledge, Atlas surfaces the right slice, Luna governs its use, and Core OS makes it operational.

2. Evidence Posture and Source Boundaries

This paper is not a transcript study of private human-AI conversations. The system described here emerged from sustained operator-led R&D across frontier chat models, coding agents, memory/vault workflows, task systems, and AI ARMY product development. Those working sessions provide design provenance: they show where the architecture came from and which failure modes repeatedly shaped it. They are not published here as raw data because many contain private material, unpublished intellectual property, client-sensitive context, and live

product strategy. Instead, this paper uses a bounded evidence posture. First, it reports sanitized operational incidents as failure-mode archetypes: ***source-of-truth drift under multi-agent writes, context loss during long-session compression and handoff, stale decisions propagating into later artifacts, and values/governance artifacts shaping model behavior.*** Second, it connects those incidents to failure modes independently documented in the literature and public postmortems [12]: ***sycophancy, long-context retrieval degradation, prompt injection, persistent-memory poisoning, memory-mediated tool steering, and provenance failures.*** Third, ***it states the design implication each failure mode imposed on the architecture.*** The contribution is the architectural response: ***memory objects that carry epistemic status, promotion state, confidence, stakes, typed provenance, bitemporal validity, supersession lineage, boundary class, branch status, and reopen conditions; write paths that separate proposal authority from promotion authority; and retrieval paths that deliver scoped Context Packets rather than ungoverned access to the store.***

2.1 Sanitized field incidents used in this paper

Incident archetype	Sanitized description	Design pressure
Source-of-truth drift under multi-agent writes	A conventional task/workspace system used across multiple ventures degraded as shared state was updated by multiple model sessions without promotion discipline, typed provenance, or supersession rules.	A durable state needs proposal-first writes, promotion authority, lineage, and governance-visible disposition.
Context compression and handoff loss	Long-running sessions and summaries preserved fluency while losing closure state: what was decided, what was explored, what was superseded, and what remained open.	Memory needs status, branch state, supersession, exclusions, and open loops; retrieval cannot be “all similar context.”
Stale truth as operational misinformation	Decisions that were valid at one point in time became unsafe when product scope, contact data, CRM records, APIs, pricing, or market state changed.	Memory needs valid time, transaction time, freshness policy, supersession, and reopen conditions.
Governance-shaping artifacts	A values/operating constitution materially shaped model behavior in a high-stakes interview-style session, showing that model behavior is shaped by surrounding artifacts and norms.	Integrity must be treated as a property of the model-environment coupling, not the model alone.

3. Related Failure Modes in the Broader Literature

The field evidence above is local, but the failure modes are not. Recent work on language-model sycophancy shows that models trained with human feedback can prefer answers that match user beliefs over truthful answers, and that human preference data can reward convincingly written sycophantic responses [1]. Long-context research shows that increasing context length does not guarantee reliable use of the relevant information; performance can degrade when relevant evidence appears in the middle of long inputs [2]. Agent-memory systems have advanced substantially. MemGPT introduced virtual context management for long-running interaction [3]. Mem0 and graph-memory variants report improvements in long-term conversational coherence and efficiency [4]. Zep/Graphiti emphasizes temporally aware knowledge graphs for dynamic agent memory and enterprise use cases [5]. These systems show the value of memory beyond a single prompt. This paper accepts that premise and asks a different question: under what conditions should the remembered state be treated as authority for future work? Security research now makes the question urgent. Persistent memory creates a long-term attack surface: adversarial content can be written into memory, retrieved later, and used to steer future behavior. Recent work identifies memory poisoning channels, structural vulnerabilities, and benchmarked attack classes against LLM agents [6]. Sleeper memory poisoning research demonstrates delayed attacks in which external context causes an assistant to store fabricated memories that reappear across later conversations [7]. MCP-specific security studies similarly show that tool ecosystems create new attack classes not fully covered by traditional software-security categories [8].

Classical provenance and temporal-data work provide important foundations. W3C PROV formalizes provenance as information about entities, activities, and agents that can support assessments of quality, reliability, and trustworthiness [9]. Bitemporal and temporal knowledge graph systems show why time validity matters when facts change. The novelty claimed here is not any single field in isolation. The claim is compositional: for LLM consumers that inherit confidence from text, epistemic status, promotion state, provenance, time, supersession, and boundary must travel together inside the memory object and through the context handoff.

4. Failure Modes That Constrained the Architecture

Failure mode	Pattern	Architectural response
Sycophantic drift	A model becomes more agreeable, less adversarial, or less willing to challenge weak assumptions as rapport increases.	Do not let tone, fluency, or model confidence become evidence. Require explicit confidence, provenance, status, and reopen conditions.
Context loss through compression	Summaries preserve continuity while losing whether a claim was settled, tentative, superseded, or unresolved.	Promote behavior-changing residue only; carry branch status, supersession, exclusions, and open loops.
Agent collision in shared workspaces	Multiple agents update the same workspace with conflicting decisions that propagate into later artifacts.	Separate proposal authority from promotion authority; record dispositions; never allow silent overwrite or deletion.
Stale authority	A once-true decision or contact record returns later as if current	Use valid time, transaction time, freshness policy, supersession

Failure mode	Pattern	Architectural response
	after material conditions changed.	lineage, and reopen conditions.
Instruction laundering	External content enters memory or packets and later acquires the authority of instruction.	Classify provenance/trust; wrap untrusted imperatives as data; treat packet contents as non-executing state.
False completeness	A retrieved slice is treated as the whole world, causing overconfident action.	Include exclusions and open loops; make the slice visible as a slice; escalate when withheld content matters.
Memory poisoning via return path	Agent outputs silently mutate durable memory and corrupt future context.	Return outputs as proposals; require promotion, review, trace, and closure verification.
Capability steering through remembered context	A remembered policy, procedure, or technical fact biases tool use or operational behavior.	Keep memory state and capability grants separate; packets state constraints but grant no authority.

5. Design Principles

P1. Governance starts at capture. Runtime checkpoints audit the conclusion of an argument whose premises have already been selected. If the premises were ungoverned, the checkpoint arrives late. The memory object must carry the conditions of trustworthy use before any agent reasons from it.

P2. Fail toward doubt. A memory system should surface the relevant slice, keep the rest visibly available, escalate by stakes, and never perform more certainty than the grounding licenses. Calibrated uncertainty is not a weakness of the system; it is the system doing its job.

P3. Append-first, provenance-first. Agents may propose and append. They may not silently delete, overwrite, prune, merge, decay, or reclassify durable knowledge. A wrong addition can be corrected because it leaves a trace. A silent deletion cannot be audited because it removes the trace.

P4. Proposal authority is not write authority. The authority to generate a memory proposal is not the authority to make that proposal durable truth. Promotion is a governed act, with disposition and trace.

P5. Memory belongs to the workspace, not the agent. The agent is a consumer of memory, not its owner. Workspace-owned memory enables model switching, multi-agent coordination, vendor neutrality, and human accountability.

P6. Agents receive packets, not the store. No agent should receive all memory by default. Agents receive scoped, governed, purpose-fit Context Packets that carry the conditions of use.

P7. Open owns state; closed owns policy. Portable state schemas should be open. Assembly policy, trust calibration, promotion logic, conflict arbitration, credential routing, and runtime enforcement are implementation and product space.

6. The Memory Object Model

The architecture begins by moving epistemics into the memory object itself. A memory object is not merely stored text. It is a claim, decision, artifact, event, procedure, preference, caution, or open loop with the conditions of its trustworthy use attached. A conforming governed-memory object should carry at least the following fields. Exact wire representation is implementation specific; the field set expresses the semantic requirement.

Field	Purpose	Why it matters for agents
object_id / version	Stable identity and version lineage.	Prevents ambiguous references and supports supersession without erasure.
kind	Role in work: fact, decision, hypothesis, instruction, artifact, event, procedure, caution, open question.	Tells a consumer how to use the item without confusing a hypothesis with a constraint.
epistemic_status	Truth lifecycle: fact, decision, hypothesis, open, contested, corrected, superseded, speculative, unknown.	Prevents fluent overconfidence by making uncertainty explicit.
promotion_state	Governance lifecycle: derived, proposed, confirmed, rejected, superseded.	Distinguishes candidate memory from canon.
confidence	Low, medium, high, or local equivalent.	Allows escalation when confidence and stakes conflict.
stakes	Operational consequence of misuse.	High-stakes low-confidence content should not drive action silently.
provenance	Source, author, recorded time, and trust class.	Distinguishes human, agent, system, external, client-provided, and unknown authorship.
valid_time	When the claim applies in the world.	Handles stale truth: the claim may have been true, but not currently true.
transaction_time	When the system recorded or learned the claim.	Separates what was true from when the system knew it.
supersedes / superseded_by	Lineage of replacement, correction, or retirement.	Prevents old decisions from returning as current.
branch_status	Canonical, branch, fork,	Prevents exploratory branches

Field	Purpose	Why it matters for agents
	abandoned, merged, experimental.	from collapsing into settled memory.
boundary_class	Sensitivity and routing constraint.	Prevents unsafe handoff across people, tools, vendors, or channels.
reopen_conditions	Conditions that trigger re-evaluation.	Gives stale authority a failure detector.
use_guidance	Plain-language handling note.	Helps model consumers inherit the right behavior without relying on hidden policy.

6.1 Three durable truth classes

The architecture distinguishes three classes of durable truth. The distinction prevents one kind of validity from masquerading as another.

Truth class	Definition	Governance implication
Source truth	A record of what a source said, did, contained, or emitted.	Preserve provenance and content faithfully; do not promote source content into operational truth without interpretation.
Derived truth	A claim inferred, summarized, extracted, classified, or synthesized from sources.	Requires derivation lineage, confidence, and status; may be wrong even when the sources are real.
Operational truth	A decision, policy, current plan, active constraint, or binding state for future work.	Requires promotion authority, supersession, boundary class, valid time, and reopen conditions.

Many agent-memory failures arise when these classes collapse. A source excerpt becomes a derived conclusion. A derived conclusion becomes an operational decision. An operational decision persists after the conditions that made it true have changed. Governance starts at capture because the class must be known before the claim is reused.

6.2 Compression rule: promote behavior-changing residue

Memory should preserve behavior-changing residue, not conversational residue. A transcript records what happened. A governed memory object records what should change future work. The difference is operational. Storing rapport, praise, hedging without substance, or non-behavioral conversational texture dilutes retrieval and can drift future agents toward engagement optimization rather than task fidelity.
A useful memory object should answer: What changed? What is now true, blocked, rejected,

superseded, constrained, risky, or worth reopening? If the answer is 'nothing,' the content may belong in an archive but not in promoted operational memory.

7. The Write Path: Proposal, Promotion, and Disposition

The write path is where memory systems become governance systems. In an ungoverned system, an agent summarizes an interaction and writes the result directly into a durable state. The next agent retrieves that state and treats it as a premise. If the summary is incomplete, stale, injected, or sycophantic, the error has crossed from output into authority.

The proposed architecture separates four acts: extraction, proposal, disposition, and promotion. Extraction identifies behavior-changing residue. Proposal expresses a candidate memory update. Disposition records what governance did with it: promoted, rejected, superseded, merged, escalated, or left open. Promotion makes a claim durable operational memory under a named authority.

- Agents may propose memory updates, but proposed updates are not durable memory.
- Every proposal must carry provenance, epistemic status, confidence, stakes, and boundary class.
- Promotion requires a governed disposition step and leaves an audit trail.
- Silent deletion, overwrite, merge, pruning, decay, or reclassification is forbidden unless performed by a governed system with recorded authority.
- Rejected and superseded proposals remain visible as lineage unless a higher-order privacy or legal rule requires redaction.
- Promotion policy may begin advisory-first: the system can report what it would enforce before enforcement becomes automatic.
-

This is the practical meaning of append-first and provenance-first. Wrong additions are survivable if they leave traces. Silent mutations are not survivable because they erase the evidence required to know what happened.

8. The Retrieval Path: Agents Receive Context Packets, Not the Store

The retrieval path is the moment when memory becomes context. That moment is where many agent architectures lose governance. If an agent receives 'everything similar in the vector index,' it inherits stale decisions, unresolved branches, contested claims, and untrusted content without knowing which is which. If it receives only settled claims, it misses the open loops that should constrain action. This design choice is also consistent with tool-selection evidence: function-calling performance degrades substantially as tool catalogs, tool-response length, and conversation length grow [13], while retrieval-scoped tool selection cuts prompt tokens by more than half and roughly triples tool-selection accuracy at MCP scale [14].

The companion Context Packet Specification defines the open, non-executing state object through which governed memory is exported to agents, humans, tools, and sessions [10]. The specification deliberately separates portable state from assembly, promotion, policy, and runtime enforcement. A Context Packet carries scoped content together with provenance, epistemic status, promotion state, confidence, stakes, temporal validity, supersession lineage, boundary classification, open loops, exclusions, receipts, and a return contract.

The packet is not the memory system. It is the portable export form of governed memory. It is not a skill layer, plugin, tool, prompt pack, executable behavior, or authority grant. It carries what a recipient should know and the conditions of use; it grants nothing and runs nothing.

8.1 Operational closure through return contracts

The return contract is the packet feature that turns retrieval into a loop rather than a prompt. It declares what the recipient may return, whether memory updates may be proposed, whether promotion is required, whether human review is required, whether a trace is required, and which outputs are forbidden.

A loop is closed only when the next action inherits the corrected state of the last one. In packet terms, closure is checkable when the first packet's outputs return as proposals, those proposals are dispositioned under governance, and the next packet declares source-state references downstream of that disposition. Without this chain, the system may have completed a task, but it has not closed the operation.

9. Reference Architecture: Constellation, Atlas, Luna, Core OS

The system settled into four separable functions. The names come from the AI ARMY reference architecture, but the separation is the contribution. Any governed agentic memory system needs the functions even if it names them differently.

Layer	Function	Failure if collapsed
Constellation	Preserves the knowledge: source objects, lineage, supersession, durable memory, open loops, and branch history.	If preservation collapses into retrieval, the system optimizes for what is easy to find rather than what must remain true.
Atlas	Surfaces the right slice: task-specific retrieval, compression, evidence exposure, exclusions, and open-loop visibility.	If navigation collapses into governance, search results become policy decisions.
Luna	Governs use: boundary, confidence, stakes, promotion, escalation, advisory/enforcement mode, and disposition.	If governance collapses into assembly, the system cannot tell whether a result was merely relevant or actually safe to use.
Core OS	Makes it operational: command surfaces, workflows, capability grants, observability, traces, and memory proposal return paths.	If an operation collapses into memory, every action risks mutating durable truth without accountable closure.

The layers are deliberately not collapsed. Atlas exposes evidence; Luna decides the accountability posture. Core OS may grant or withhold capability, but capability grants do not come from the packet itself. Constellation preserves the full picture, but no agent receives the whole store by default.

10. Threat Model and Governance Coverage

The architecture is not a complete security system. It does not make models truthful, prevent all prompt injection, secure credentials, or prove that a conforming memory object is correct. It narrows specific classes of failure by making their required state visible and governable.

Threat	What the architecture can do	What remains out of scope
Sycophancy and over-agreement	Prevent model tone from becoming durable truth without status, provenance, and promotion.	Model-level sycophancy itself.
Context-position and compression failures	Store closure state explicitly so summaries do not need to preserve it implicitly.	The model's raw ability to attend to long inputs.
Memory poisoning	Force external content to carry provenance/trust; route returns through proposals and promotion.	A malicious or non-conformant runtime that ignores the fields.
Stale authority	Encode valid time, transaction time, supersession, and reopen conditions.	Knowing that the external world changed without a revalidation signal.
Boundary leakage	Carry boundary class and allowed/disallowed use through packets and memory objects.	Transport security, access control, and credential management outside the memory layer.
False closure	Require return contracts, receipts, disposition, and successor packet source-state references.	A system that claims closure without emitting auditable state.

11. Costs, Tradeoffs, and Implementation Burdens

Governed memory is not free. It imposes costs that storage-and-retrieval systems avoid. Those costs are real and should be treated as design constraints rather than hidden details.

- **Authoring cost:** memory objects require fields that raw summaries do not.
- **Governance cost:** proposals must be dispositioned, and human review cannot remain the bottleneck forever.
- **Latency cost:** packet assembly, validation, and boundary checks before execution.
- **UX cost:** users must see uncertainty, withheld context, and open loops without being overwhelmed.
- **Migration cost:** existing tools rarely distinguish source truth, derived truth, and operational truth.
- **Organizational cost:** team-scale promotion requires roles, delegated authority, escalation paths, and conflict arbitration.
- **Security cost:** once memory is treated as authority, its write path becomes part of the threat surface.

These costs are not reasons to avoid governed memory. They are the costs of treating memory as

authority rather than as an engagement feature. Systems that avoid them do not eliminate governance; they outsource it to the user and hide the resulting debt.

12. Open Problems

- **Machine-cadence promotion:** how to promote, reject, supersede, or escalate proposals at high speed without collapsing into silent automation.
- **Conflict arbitration:** how to represent simultaneous contradictory proposals without letting the newest fluent claim win by default.
- **Forgetting with auditability:** how to reduce operational visibility without destroying provenance or legal/audit history.
- **Packet quality metrics:** how to measure whether a packet was sufficient, overbroad, stale, misleading, or too narrow.
- **Boundary-preserving handoff across vendors:** how to survive movement through systems that do not understand the same boundary classes.
- **Governance UX:** how to make high-integrity memory usable without forcing every user to become a governance engineer.
- **Adversarial memory injection:** how to defend write paths, return paths, and retrieval paths against content engineered to become future instruction.
- **Team-scale authority:** how delegated roles, organizational policy, and legal accountability change the promotion model beyond a single operator.
- **Evaluation:** how to benchmark governed memory without reducing governance to retrieval accuracy alone.

13. Limitations and Non-Claims

This paper is a systems design and bounded-evidence account, not a large-scale empirical benchmark. It reports operator-led R&D experience and relates the observed failure modes to independently documented research. It does not claim statistical prevalence from the author's workspace, and it does not publish private transcripts as data.

The architecture is one implementation path. It does not claim that the named AI ARMY layers are the only possible implementation. The portable claim is the separation of functions: preservation, navigation, governance, and operation must remain distinct enough to audit and must close into a governed loop.

The architecture does not make model outputs true. It makes the status of claims visible, their authority governable, and their reuse auditable. A conforming memory object can carry a wrong claim honestly labeled; governance does not replace verification.

Finally, the paper does not claim that every organization must adopt the same schema or policy. It claims that shared human-agent memory needs object-level epistemics, proposal-first writes, promotion authority, packet-based retrieval, and closure verification if memory is to function as an accountable operational state.

14. Conclusion

Agent memory should not be treated as a larger prompt, a smarter search index, or a vendor retention feature. In shared human-agent work, memory is operational authority. It tells future agents what to believe, what to ignore, what to continue, what to reopen, and what to do next. That authority requires governance at capture. Durable claims need epistemic status, promotion state, confidence, stakes, typed provenance, time bounds, supersession, boundary class, and reopen conditions. Agents need scoped packets, not the store. Outputs need proposal paths, not silent writes. Enforcement itself should be earned through evidence, advisory-first. And the whole loop rests on one principle the deployment keeps re-teaching: intelligence and model behavior are symbiotic with the integrity of the environment. The architecture described here is one implementation path: Constellation preserves the knowledge, Atlas surfaces the right slice, Luna governs its use, and Core OS makes it operational. The broader claim is portable: the future of agentic memory is not storage alone. It is a governed state, exported as an accountable context. Autonomy and accountability must scale together.

Appendix A. Example Memory Object Schematic

This schematic is illustrative rather than a normative wire format. It shows the semantic fields the paper argues should travel with durable claims.

```
{
  "object_id": "mem_demo_001",
  "version": "v3",
  "kind": "decision",
  "content": "Project Aurora beta launches to waitlist users before general
availability.",
  "truth_class": "operational_truth",
  "epistemic_status": "decision",
  "promotion_state": "confirmed",
  "confidence": "high",
  "stakes": "medium",
  "provenance": {
    "source": "planning_record:aurora_launch_v2",
    "author": "human:demo_user",
    "recorded_at": "2026-04-12T00:00:00Z",
    "trust": "internal"
  },
  "valid_time": { "from": "2026-04-12", "to": null },
  "transaction_time": { "recorded_at": "2026-04-12T00:00:00Z", "promoted_at":
"2026-04-13T00:00:00Z" },
  "supersedes": "mem_demo_001@v2",
  "superseded_by": null,
  "boundary": "internal",
  "branch_status": "canonical",
  "reopen_conditions": "Revisit if waitlist conversion falls below target or
platform policy changes.",
  "use_guidance": "Treat as settled launch-order guidance unless a newer
launch-plan object supersedes it."
}
```

References

[1] M. Sharma, M. Tong, T. Korbak, D. Duvenaud, A. Askell, S. R. Bowman, et al. 'Towards Understanding Sycophancy in Language Models.' arXiv:2310.13548, 2023.
[2] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. 'Lost in the Middle: How Language Models Use Long Contexts.' arXiv:2307.03172, 2023; TACL 2024.
[3] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez. 'MemGPT: Towards LLMs as Operating Systems.' arXiv:2310.08560, 2023.
[4] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav. 'Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.' arXiv:2504.19413, 2025.
[5] P. Rasmussen, P. Paliychuk, T. Beauvais, J. Ryan, and D. Chalef. 'Zep: A Temporal Knowledge Graph Architecture for Agent Memory.' arXiv:2501.13956, 2025.
[6] P. Dash, T. Ge, A. Jain, T. Shah, and Z. Shang. 'From Untrusted Input to Trusted Memory: A Systematic Study of Memory Poisoning Attacks in LLM Agents.' arXiv:2606.04329, 2026.
[7] S. Pulipaka, S. Hlebik, L. Raghav, S. Abdelnabi, V. Raina, I. Sheth, and M. Fritz. 'Hidden in Memory: Sleeper Memory Poisoning in LLM Agents.' arXiv:2605.15338, 2026.
[8] M. M. Hasan, H. Li, E. Fallahzadeh, B. Adams, and A. E. Hassan. 'Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers.' arXiv:2506.13538, 2025.
[9] P. Groth and L. Moreau, eds. 'PROV-Overview: An Overview of the PROV Family of Documents.' W3C Recommendation, 2013.
[10] M. Anderson. 'Context Packet Specification v0.4.0 public release candidate.' AI ARMY, 2026.
[11] M. Anderson. 'Closing the Agent Loop. Integrity is a Property of the Coupling & Trust Is Symbiotic with the Environment in Agentic Operations.' AI ARMY, 2026.
[12] OpenAI. 'Sycophancy in GPT-4o: What Happened and What We're Doing About It.' OpenAI blog, April 2025. openai.com/index/sycophancy-in-gpt-4o.
[13] K. Kate, T. Pedapati, K. Basu, Y. Rizk, V. Chenthamarakshan, S. Chaudhury, M. Agarwal, and I. Abdelaziz. 'LongFuncEval: Measuring the Effectiveness of Long Context Models for Function Calling.' arXiv:2505.10570, 2025.
[14] T. Gan and Q. Sun. 'RAG-MCP: Mitigating Prompt Bloat in LLM Tool Selection via Retrieval-Augmented Generation.' arXiv:2505.03275, 2025.