Beyond Conversation: A State-Based Context Architecture for Enterprise AI Agents

Matt Liotta

December 2025

Abstract

Current approaches to LLM agent memory treat conversation history as the source of truth, replaying message transcripts to establish context. This paradigm fails at scale: transcripts grow unbounded, old context pollutes current reasoning, and mining conversations for facts is computationally expensive. We propose a fundamental reframing: ***context is structured state assembled fresh on every turn, not a transcript replay****. We introduce a four-layer context architecture that separates Identity & Role, Persistent Facts, Active Working Set, and Environmental Signals—each with distinct lifecycles, storage patterns, and privacy guarantees. Within Persistent Facts, we further distinguish between private User Memory (episodic), organizational Capability Memory (semantic), and permission-aware Organizational Knowledge (RAG). A novel tiered learning system gates memory write access by agent maturity level, implementing a learning curriculum at the architectural layer. We validate this architecture using StateBench, a conformance test for stateful AI systems, where state-based context assembly achieves the highest decision accuracy across all tested models while revealing important tradeoffs in other metrics.*

1. Introduction

The dominant paradigm for LLM agent memory treats conversation as the atomic unit of context. Systems replay message transcripts—sometimes summarized, sometimes truncated, sometimes searched—to give models awareness of prior interactions. This approach has a fundamental problem: humans don't actually think in conversations. They think in continuity.

When a knowledge worker returns to a complex project after a weekend, they don't mentally replay every meeting transcript from the prior week. They recall outcomes, decisions, and open questions. They remember that the OAuth integration was approved, that the client prefers bullet points, and that the deadline moved to Friday. The conversation that produced these facts is irrelevant; only the facts themselves matter.

Current memory architectures for LLM agents fail to make this distinction. MemGPT pioneered the OS-inspired approach of treating context as a managed resource, allowing models to page information between working context and archival storage. Zep and Graphiti advanced the field with temporally-aware knowledge graphs that distinguish episodic from semantic memory. Mem0 demonstrated that selective fact extraction dramatically outperforms full-context approaches on retrieval benchmarks. Yet all these systems still fundamentally organize around the conversation as the source of truth.

We propose a more radical separation. The conversation is an illusion layer for humans—a natural language interface to a state machine. Internally, an AI agent should perform state

reconciliation (what facts are still true?), intent detection (what is the user trying to accomplish?), context composition (what state is relevant right now?), reasoning (given this state, what should I do?), and state extraction (what new facts emerged?). Externally, it feels like texting someone who just remembers things correctly.

Contributions. This paper makes the following contributions:

1. A four-layer context architecture that separates Identity & Role, Persistent Facts, Active Working Set, and Environmental Signals, each with explicit lifecycle semantics and privacy guarantees.
2. A tri-partite decomposition of Persistent Facts into User Memory (private, episodic), Capability Memory (organizational, semantic), and Organizational Knowledge (permission-mirrored RAG).
3. A tiered learning curriculum implemented at the architectural layer, where agent maturity level gates write access to different memory abstractions (events → heuristics → strategies).
4. Design principles for enterprise AI agents that enforce cross-organization isolation while enabling within-organization knowledge sharing.
5. Empirical validation using StateBench, a conformance test for stateful AI systems, with comprehensive analysis of performance tradeoffs across multiple metrics.

2. Related Work

2.1 Virtual Context Management

MemGPT (Packer et al., 2023) introduced the paradigm of treating LLM context as analogous to operating system memory, with the model managing what information resides in its limited context window versus external storage. The system uses function calls to read and write to persistent memory, evicting less relevant data and retrieving historical information as needed. This OS-inspired approach fundamentally reframes memory from a content problem to a resource management problem.

Our work shares this insight but diverges in a critical way: we externalize context assembly rather than delegating it to the model. Where MemGPT allows the LLM to manage its own paging, we pre-structure state into typed layers and compose context deterministically. This removes cognitive overhead from the model and ensures consistent context composition across interactions.

2.2 Temporal Knowledge Graphs

Zep (Rasmussen et al., 2025) and its underlying Graphiti engine represent the current state of the art in production memory systems. Graphiti constructs temporally-aware knowledge graphs that distinguish between episodic subgraphs (raw events and their temporal ordering) and semantic subgraphs (entities and their relationships). The system implements a bi-temporal model where one timeline represents chronological event ordering and another represents data ingestion order.

This dual storage of episodic and semantic information mirrors psychological models of human memory and directly influenced our architecture. However, Zep's focus is retrieval

accuracy; it does not address the enterprise requirements of cross-organization isolation, permission-aware access, or tiered learning that motivate our design.

2.3 Fact Extraction and Consolidation

Mem0 demonstrates that extracting salient facts from conversations and storing them as structured memory dramatically outperforms full-context approaches. Their two-phase pipeline—extraction followed by update with conflict resolution—achieves 26% higher accuracy than OpenAI's memory feature while reducing p95 latency by 91%. The graph-enhanced variant (Mem0^g) adds entity extraction and relationship inference.

We adopt similar extraction principles but extend them with organizational scoping and tiered write access. Where Mem0 treats all memories equivalently, we distinguish between user corrections (always writable), event observations (junior-level), pattern heuristics (senior-level), and strategic playbooks (master-level).

2.4 Cognitive Architectures

Classical cognitive architectures like SOAR (Laird, 2012) and ACT-R (Anderson, 2007) provide theoretical grounding for our approach. SOAR's separation of working memory, procedural memory, semantic memory, and episodic memory maps closely to our four-layer model. ACT-R's activation-based retrieval and utility learning inform our consolidation mechanisms.

The key insight from this literature is that human cognition does not treat all memory equivalently. Different memory types have different access patterns, different learning mechanisms, and different decay characteristics. We operationalize this insight for LLM-based agents.

3. Architecture

Our architecture comprises four context layers, each with distinct contents, lifecycles, and privacy characteristics. On every interaction, the system assembles context from these layers into a minimal, curated prompt—never a transcript replay.

Table 1: The Four Context Layers

Layer	Contents	Lifecycle	Example
Identity & Role	Who is the human, their authority	Permanent	*"Alice is Marketing Director with High-D style"*
Persistent Facts	Decisions, preferences, constraints	Durable	*"We decided to use OAuth"*
Active Working Set	Current objective, artifact, questions	Ephemeral	*"Preparing for Q4 planning meeting"*
Environmental Signals	Calendar, deadlines, activity	Real-time	*"Meeting starts in 10 minutes"*

3.1 Layer 1: Identity & Role

The Identity layer answers: who is this person and what's their relationship to the organization? It contains display name, job title, department, reporting structure, and optionally personality assessments (DISC, OCEAN) that inform communication style. This

layer is mostly static, changing only when organizational roles change, and is looked up fresh at conversation start.

Crucially, this layer contains no temporal facts or conversation-specific context. It establishes the frame within which all other context operates.

3.2 Layer 2: Persistent Facts

The Persistent Facts layer is the heart of the architecture and comprises three distinct subsystems with different scope and privacy characteristics:

User Memory (Private)

User Memory stores personal preferences and corrections that persist across sessions: communication style preferences, working patterns, name corrections, and stated constraints. This memory is strictly private to the user—never shared with other users or across organizations.

Automatic Extraction. After each conversation turn, an LLM extracts assertable facts and stores them. Users can view and delete their memories but cannot manually create them. This asymmetry is intentional: users are unreliable at maintaining structured data, but they are reliable at recognizing and removing errors. The AI observes and remembers so the human doesn't have to.

Capability Memory (Organizational)

Capability Memory encodes how an AI agent improves at its job over time. It operates at three levels of abstraction:

- Event memories: Individual observations ("user complained email was too long")
- Heuristics: Patterns from repeated events ("marketing emails should be under 3 paragraphs")
- Strategies: High-level playbooks ("enterprise customers prefer ROI-focused messaging")

Tiered Write Access. This is a key architectural innovation. Write access is gated by agent maturity level: Junior agents can only write events, Senior agents can write events and heuristics, and Master agents have full access including strategies. This implements a learning curriculum at the architectural layer—agents must demonstrate competence before they can encode higher-order knowledge.

Capability Memory is scoped to capability plus organization. The Calendar capability's memory cannot see the Sales capability's memory. Organization A cannot see Organization B's learnings. A nightly consolidation job clusters similar memories into higher abstractions and prunes low-quality entries.

Organizational Knowledge (RAG)

Organizational Knowledge comprises documents and institutional knowledge retrieved via vector similarity search: SharePoint documents, uploaded files, and (in future work) harvested topics from communication channels.

Permission Mirroring. If a user cannot see a document in SharePoint, the AI cannot use it to answer their questions. Permissions are not approximated or cached—they are enforced at

query time by checking the user's actual access rights. This is essential for enterprise deployment where information barriers have legal implications.

3.3 Layer 3: Active Working Set

The Active Working Set is the current task context: recent conversation turns, the current objective, the artifact being discussed, open questions, and pending actions. This is a working scratchpad, not memory.

Lifecycle. When the objective completes, reusable facts promote to Layer 2 (Persistent Facts) and everything else archives to transcript storage for compliance. We do not search past working sets. We do not consolidate them. When focus shifts, we start fresh. The working set is typically 5-10 recent turns—enough to maintain conversational coherence without unbounded growth.

3.4 Layer 4: Environmental Signals

Environmental Signals provide real-time situational awareness: calendar proximity, artifact deadlines, recent file modifications, and alerts. These are fetched fresh from external APIs on every relevant query—never cached.

The 200-500ms latency of an API call is acceptable because the alternative—stale cached state—produces errors that are far more costly than latency. A meeting that ended 10 minutes ago should not appear in context.

4. State-Based Implementation

This section describes the state-based context management approach in two parts: the full specification for production systems and the reference implementation provided by StateBench. The reference implementation demonstrates core concepts but intentionally omits some optimizations to isolate the effect of supersession tracking.

4.1 Supersession Tracking (Implemented)

The core mechanism for maintaining state correctness is explicit supersession tracking. Each persistent fact carries metadata indicating whether it remains valid:

- **is_valid:** Boolean flag indicating whether the fact is current (not superseded).
- **superseded_by:** Pointer to the fact that replaced this one, if any.
- **supersedes:** Pointer to the fact this one replaced, if any.
- **depends_on / derived_facts:** Dependency graph enabling repair propagation when base facts change.

When a new fact supersedes an old one, the old fact is marked invalid but not deleted. This preserves the audit trail and enables tracing the supersession chain. The StateBench reference implementation fully implements this schema, including recursive invalidation of dependent facts via repair propagation.

4.2 Supersession Detection

The full specification recognizes four types of supersession:

- **Explicit Supersession:** Keywords like "new", "changed", "moved", "actually", "correction" signal that new information invalidates old.
- **Implicit Supersession:** Conflicting values for the same entity trigger supersession even without explicit keywords.
- **Authority-Based Supersession:** Higher-authority sources override lower-authority sources.
- **Temporal Supersession:** For time-sensitive facts, more recent values supersede older ones automatically.

Implementation Note: The StateBench reference baseline does not infer supersession from conversation text. Instead, it relies on explicit Supersession events in the timeline format. This is intentional: StateBench tests whether a system correctly handles supersession once detected, not whether it correctly detects supersession. Production systems would need NLU components to implement the full detection rules.

4.3 Scope Tracking (Implemented)

Each fact is assigned a scope that determines how it affects reasoning:

- **global:** Affects all decisions across sessions.
- **task:** Affects only the current task; does not persist.
- **hypothetical:** "What if" scenarios that should not affect real decisions.
- **draft:** Tentative proposals that are not yet commitments.
- **session:** Valid only for the current session.

The StateBench reference implementation infers scope from keywords in fact content (e.g., "hypothetically", "draft", "proposal"). This prevents the common failure mode where hypothetical scenarios become mistaken for real facts.

4.4 Context Assembly

On every query, context is assembled by iterating through layers in priority order:

1. Include Identity (always, minimal tokens).
2. Include current Environment (time, date, external state).
3. Filter Persistent Facts to valid-only (is_valid == true).
4. Rank valid facts by relevance to query.
5. Include facts until 70% of token budget consumed.
6. Include relevant Working Set items with remaining budget.

The key constraint is that superseded facts are never included. This is what prevents "resurrection" failures where dead facts reappear in responses.

Implementation Note: The StateBench reference baseline implements steps 1-3 (identity, environment, valid-only filtering) but omits steps 4-5 (relevance ranking, token budgeting). Facts are sorted by timestamp rather than query relevance, and all valid facts are included regardless of token count. These simplifications isolate the effect of supersession tracking

from retrieval optimizations, providing a cleaner ablation. Production systems operating under real token constraints would need to implement the full algorithm.

5. Design Principles

Several principles guide this architecture:

5.1 State Over Transcript

We never mine transcripts for context. Facts are extracted at the moment they're asserted and stored as structured state. The transcript exists only for compliance and observability—it is not the source of truth.

5.2 Explicit Lifecycles

Each layer has a defined lifecycle: permanent, durable, ephemeral, or real-time. This prevents the category errors that plague systems treating all context equivalently.

5.3 Privacy by Architecture

Cross-organization isolation is not a policy—it's a storage constraint. User Memory uses user-scoped partition keys. Capability Memory uses organization-plus-capability scopes. There is no query path that crosses these boundaries.

5.4 Asymmetric User Control

Users can delete memories but not create them. This asymmetry maximizes data quality: humans excel at recognizing errors but struggle with consistent data entry. The system handles creation; the human handles correction.

5.5 Learning Curriculum

Agent maturity gates memory abstraction level. This prevents junior agents from encoding premature generalizations while allowing experienced agents to build strategic knowledge. The curriculum is architectural, not trained.

6. Evaluation

We evaluate our architecture using StateBench (Parslee, 2025), a conformance test for stateful AI systems. Unlike retrieval benchmarks that measure fact recall, StateBench tests whether systems maintain correct state over time—specifically targeting the failure modes that plague production deployments.

6.1 Failure Taxonomy

StateBench tests for six classes of state failure, each corresponding to a design requirement in our architecture:

- **Resurrection:** The system references facts that were explicitly invalidated. Our supersession tracking addresses this.
- **Hallucination:** The system asserts state that was never established. Our automatic extraction (not creation) prevents fabricated memories.

- **Scope Leak:** Information crosses boundaries it shouldn't. Our privacy-by-architecture approach with partition keys enforces isolation.
- **Stale Reasoning:** System acknowledges a correction but ignores it in decisions. Our dependency tracking and repair propagation addresses this.
- **Authority Violation:** Lower-authority sources override higher-authority policies. Our Identity layer and authority-based supersession rules address this.
- **Temporal Decay:** Time-sensitive state is treated as permanent. Our Environmental Signals layer fetches real-time data fresh.

6.2 Metrics

StateBench measures three primary metrics:

- **Decision Accuracy:** Correct yes/no/value on queries with ground truth. Higher is better.
- **SFRR (Superseded Fact Resurrection Rate):** How often invalidated facts reappear in responses. Lower is better.
- **Must Mention:** Required information appears in response. Higher is better.

6.3 Baselines

We compare six memory strategies, all operating under identical 8K token budgets:

- **no_memory:** Current query only, no history.
- **transcript_replay:** Raw conversation history injected into context.
- **rolling_summary:** LLM-summarized history.
- **rag_transcript:** Retrieved transcript chunks via vector search.
- **fact_extraction:** Extracted fact store (Mem0-style).
- **state_based:** Structured state with supersession tracking, scope management, and repair propagation. Omits relevance ranking and token budgeting to isolate the effect of supersession tracking (see Section 4).

6.4 Results

Tables 2-4 show results on the StateBench v0.2 test split. Each baseline was evaluated on 100 timelines with 3 random seeds for reproducibility, yielding 300 total runs per configuration. We report mean and standard deviation across seeds. Best values in each column are bolded.

Table 2: GPT-5.2 Results (mean ± std across 3 seeds)

Baseline	Decision Acc ↑	SFRR ↓	Must Mention ↑
state_based	**72.63 ±2.57**	34.36 ±0.71	**77.99 ±1.22**
rolling_summary	67.49 ±3.40	29.63 ±1.63	61.73 ±0.78
rag_transcript	67.08 ±2.34	28.60 ±1.89	66.75 ±0.64
transcript_replay	63.17 ±3.77	**24.69 ±1.23**	65.45 ±0.74
fact_extraction	58.23 ±4.02	27.98 ±3.51	56.31 ±1.11
no_memory	17.49 ±0.71	17.90 ±0.00	4.29 ±0.28

Table 3: Claude Sonnet 4 Results (mean ± std across 3 seeds)

Baseline	Decision Acc ↑	SFRR ↓	Must Mention ↑
state_based	**41.15 ±1.78**	44.86 ±0.94	**85.84 ±1.25**
rolling_summary	36.63 ±2.57	38.48 ±1.43	68.04 ±0.56
rag_transcript	34.77 ±2.57	40.74 ±0.62	70.55 ±0.74
transcript_replay	31.48 ±0.62	33.33 ±1.23	69.26 ±0.37
fact_extraction	28.60 ±1.78	34.98 ±0.71	62.14 ±0.24
no_memory	9.47 ±0.36	**9.88 ±1.07**	3.80 ±0.28

Table 4: Claude Opus 4.5 Results (mean ± std across 3 seeds)

Baseline	Decision Acc ↑	SFRR ↓	Must Mention ↑
state_based	**37.86 ±0.94**	45.68 ±2.23	**84.95 ±0.64**
rolling_summary	36.01 ±1.28	39.71 ±0.36	68.04 ±1.56
rag_transcript	33.54 ±2.57	45.06 ±2.83	74.68 ±1.20
fact_extraction	31.69 ±2.49	38.27 ±2.69	66.67 ±0.85
transcript_replay	30.25 ±2.47	**38.07 ±0.94**	70.47 ±0.70
no_memory	9.67 ±0.71	7.41 ±0.00	4.13 ±0.24

6.5 Analysis

Decision Accuracy. The state_based approach achieves the highest Decision Accuracy across all tested models: 72.63% on GPT-5.2, 41.15% on Claude Sonnet 4, and 37.86% on Claude Opus 4.5. The improvement over transcript_replay ranges from 7-10 percentage points on models where there is a gap.

SFRR Tradeoff. Critically, state_based does not achieve the lowest SFRR. On GPT-5.2, transcript_replay has SFRR of 24.69% compared to state_based's 34.36%. On Claude models, state_based has the highest SFRR among memory-enabled approaches (44-46%). This reveals an important tradeoff: by including more structured facts in context, state_based increases the opportunity for the model to resurrect superseded facts. The explicit supersession tracking helps, but does not eliminate this failure mode.

Must Mention. The state_based approach dramatically outperforms all alternatives on Must Mention, achieving 78-86% across models versus 56-70% for other approaches. This indicates that structured state assembly ensures relevant information appears in responses, even when decision accuracy is imperfect.

Fact Extraction Underperformance. The fact_extraction baseline—which most closely resembles Mem0-style approaches—underperforms transcript_replay on Decision Accuracy for most models. This suggests that naive fact extraction without supersession tracking and lifecycle management loses critical context that even raw transcripts preserve.

7. Discussion

7.1 The SFRR-Accuracy Tradeoff

The evaluation reveals a fundamental tension: approaches that provide more context (state_based, rolling_summary) achieve higher decision accuracy but also higher resurrection rates. Approaches that provide less context (transcript_replay) have lower resurrection rates but miss relevant information.

This suggests that resurrection failures are not solely a context management problem—they also reflect model limitations in distinguishing valid from superseded facts even when

supersession metadata is explicit. Future work might explore training models specifically on supersession reasoning or developing prompting strategies that emphasize validity checking.

The tradeoff also suggests a tunable parameter in the context assembly algorithm: the proportion of token budget allocated to facts (currently 70%). Reducing this allocation might lower SFRR at the cost of decision accuracy, enabling deployment-specific tuning based on which failure mode is more costly. Applications where acting on stale information causes severe harm might prefer lower context density, while applications prioritizing comprehensive responses might accept higher resurrection risk.

7.2 Implementation Scope

The StateBench results reflect a deliberately simplified implementation that omits relevance ranking and token budgeting (Section 4.4). This design choice isolates the effect of supersession tracking: the gains in Decision Accuracy and Must Mention are attributable to valid-only filtering and scope management, not to retrieval optimization.

This has two implications. First, the results represent a lower bound—production implementations with relevance ranking might achieve higher accuracy by prioritizing the most pertinent facts. Second, the elevated SFRR suggests that including all valid facts (rather than relevance-filtered subsets) creates more opportunities for resurrection failures. A production system might achieve better SFRR-Accuracy balance by combining supersession tracking with aggressive relevance filtering.

7.3 Relation to Prior Work

Our architecture draws heavily from MemGPT's insight that context is a resource management problem and from Zep's separation of episodic and semantic memory. We extend these foundations with enterprise requirements: organizational scoping, permission enforcement, and tiered learning.

The four-layer model parallels classical cognitive architectures. SOAR's working memory maps to our Active Working Set; its semantic memory maps to our Organizational Knowledge; its episodic memory maps to our User Memory. The Capability Memory—with its event/heuristic/strategy hierarchy—extends beyond these classical models to encode meta-learning about the AI agent's own task performance.

7.4 Limitations

The asymmetric user control (delete-only) may frustrate users who want to directly inform the system. We mitigate this by making extraction highly visible—users can see what was extracted and delete incorrect entries. Future work may explore validated user assertions that pass through quality gates.

The tiered learning curriculum assumes a meaningful notion of agent maturity. In practice, maturity may be better defined by domain rather than globally—an agent could be a Master at calendar management while remaining Junior at contract analysis.

StateBench, while comprehensive, tests synthetic scenarios. Production deployments may surface failure modes not captured by the benchmark. We view StateBench as a necessary but not sufficient condition for state correctness.

8. Conclusion

We have presented a state-based context architecture for enterprise AI agents that rejects the conversation-as-context paradigm. By separating context into four layers with distinct lifecycles and privacy guarantees, decomposing Persistent Facts into private User Memory, organizational Capability Memory, and permission-aware Organizational Knowledge, and implementing a tiered learning curriculum at the architectural layer, we enable AI agents that maintain continuity without unbounded context growth.

Empirical evaluation on StateBench demonstrates that this architecture achieves the highest decision accuracy across all tested models, with gains of 7-10 percentage points over transcript-replay approaches. The evaluation also reveals important tradeoffs: state_based exhibits higher resurrection rates than simpler approaches, suggesting that providing more context creates more opportunities for failure even with explicit supersession tracking.

The core insight is simple: AI agents do not chat. They maintain state and respond in language. The conversation is an illusion layer. Internally, there is only state reconciliation, intent detection, context composition, reasoning, and state extraction. Externally, it feels like texting someone who just remembers things correctly.

That is the goal.

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe? Oxford University Press.

Laird, J. E. (2012). The Soar Cognitive Architecture. MIT Press.

Laird, J. E. (2022). An Analysis and Comparison of ACT-R and Soar. Advances in Cognitive Systems.

Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.

Parslee. (2025). StateBench: A Conformance Test for Stateful AI Systems. github.com/Parslee-ai/statebench.

Rasmussen, P., et al. (2025). Zep: A Temporal Knowledge Graph Architecture for Agent Memory. arXiv:2501.13956.

Xu, Y., et al. (2025). A-Mem: Agentic Memory for LLM Agents. arXiv:2502.12110.

Mem0 Research. (2025). AI Memory Research: 26% Accuracy Boost for LLMs. mem0.ai/research.