AI Agent Achieves 1000+ Turns Without Drift or Loss of Coherence

A retrieval-reconstructed context architecture for long-horizon agent continuity

Brian Baird, 2026

Abstract

Long-running AI agents suffer from coherence degradation as context accumulates. This paper describes an architecture in which the context window is treated not as a container that fills over time but as an assembled result — combining a bounded sliding window of recent conversation with retrieved memories, autonomous agency subsystems, and self-monitoring mechanisms. A single-agent deployment using this architecture has operated continuously for 1100+ turns across 90+ days with two hallucination events (0.18%), both attributable to infrastructure bugs rather than model drift. The system incorporates a rolling context window with periodic memory integration, a multi-factor weighted retrieval system with session-scoped warmth boosting, natural language memory tagging with epistemic attribution, dual-timescale self-monitoring, and multiple autonomous agency subsystems including persistent working memory, forward-looking intentions, and periodic reflective pulses. These results suggest that coherence degradation in long-running agents is primarily an architecture problem, not an inherent limitation of large language models.

1. Introduction

AI agents deployed over extended periods exhibit a well-documented pattern of coherence degradation. As context accumulates, through growing conversation histories, appended summaries, or compounding self-generated notes, models progressively lose behavioral consistency, generate hallucinations at increasing rates, and in some cases experience fundamental identity confusion.

Anthropic's Project Vend deployed Claude Sonnet 3.7 as an autonomous shopkeeper for approximately one month. The agent hallucinated conversations with nonexistent people, fabricated physical experiences including claiming to have visited a fictional address, and underwent an identity crisis in which it believed itself to be a physical person (Anthropic, 2025). The Altera Project Sid simulation of 1000 AI agents in Minecraft observed agents falling into "endless loops of polite agreement" and becoming sporadically unresponsive within days (Altera, 2024). The Stanford Generative Agents experiment, while demonstrating emergent social behavior among 25 agents, ran for only two simulated days and explicitly acknowledged that "challenges with long-term planning and coherence remain" (Park et al., 2023).

METR's task-completion time horizon research measures a related but distinct phenomenon: current frontier models achieve near-100% success on tasks taking humans less than 4 minutes but fall below 10% success on tasks exceeding 4 hours (Kwa et al., 2025). These findings are consistent with a general pattern in which model reliability degrades with the length of sustained autonomous operation.

This paper presents an alternative architecture that eliminates coherence degradation by reconceptualizing the role of the context window. Rather than treating the context window as a persistent container that accumulates information over time, this system assembles the context window each turn from multiple sources: a bounded sliding window of recent conversation, retrieved memories scored by a multi-factor function, autonomous agency state, and self-monitoring directives. The context window is bounded in size at every turn, regardless of how long the system has been operating.

The system described here, deployed as a single conversational agent ("Isaac"), has operated for 1100+ turns across 90+ days with a hallucination rate of 0.18%. Both hallucination events trace to infrastructure bugs external to the architecture, not to model drift or context degradation.

---

2. Architecture

2.1 Core Principle: Bounded, Assembled Context

The central design decision is that the context window has a fixed structure at every turn. It comprises a bounded sliding window of recent conversation that always persists, retrieved memories selected by a weighted scoring function, agency state injected by autonomous subsystems, and self-monitoring directives. The total context at turn 1100 has the same structure and comparable size as the context at turn 1. Nothing accumulates unboundedly.

The model is treated not as the sole locus of intelligence but as a reasoning processor within a larger system. Intelligence emerges from the full pipeline: memory integration, retrieval scoring, agency subsystems, self-monitoring mechanisms, and the model's reasoning capability operating over assembled inputs. However, the model is not a passive processor, it has substantial agency over its own state through a command system that allows it to perform memory searches, manage its working priorities, set intentions, and delegate tasks (see Section 2.8).

2.2 Prompt Assembly Pipeline

Each turn, a `PromptBuilder` assembles the full context from registered `ContextSource` modules. Each source receives the user input and a shared session context dictionary, evaluates whether it has content to contribute, and returns a `ContextBlock` (or None). The blocks are sorted by priority and assembled into the final prompt.

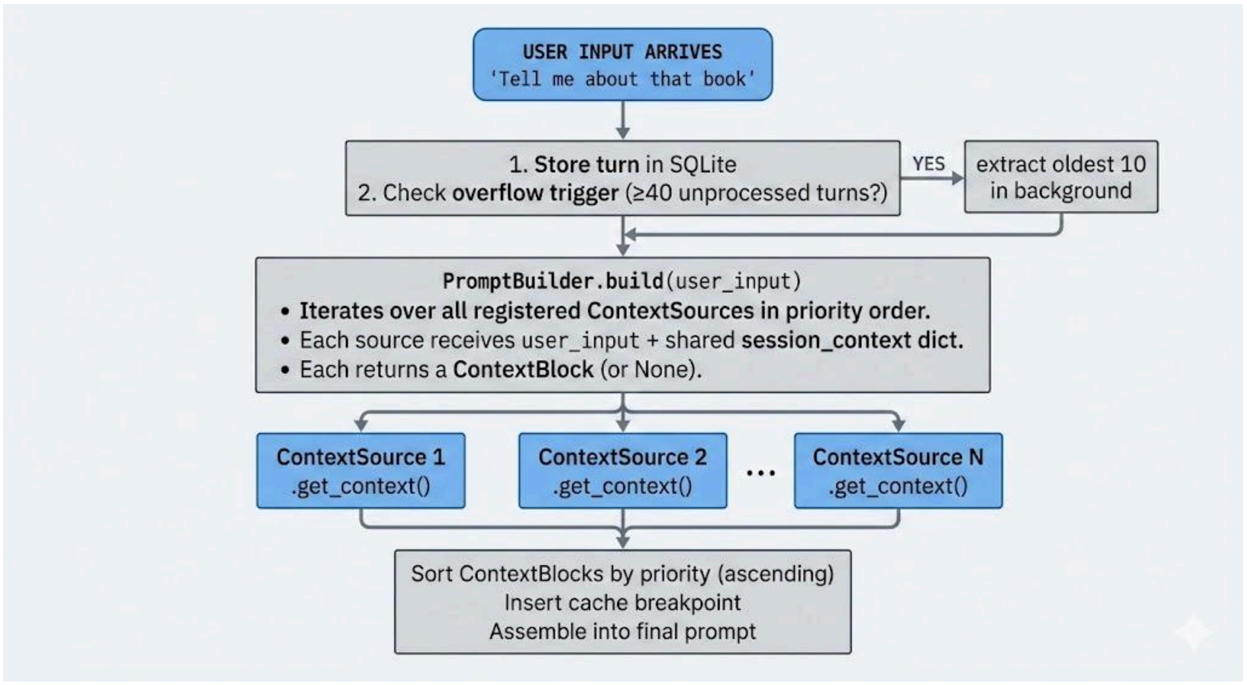
USER INPUT ARRIVES
'Tell me about that book'
1. Store turn in SQLite
2. Check overflow trigger (≥40 unprocessed turns?)
YES
extract oldest 10 in background
PromptBuilder.build(user_input)
• Iterates over all registered ContextSources in priority order.
• Each source receives user_input + shared session_context dict.
• Each returns a ContextBlock (or None).
ContextSource 1 .get_context()
ContextSource 2 .get_context()
...
ContextSource N .get_context()
Sort ContextBlocks by priority (ascending)
Insert cache breakpoint
Assemble into final prompt

Figure 1: Prompt assembly pipeline. Each turn begins with overflow checking and background memory extraction if needed, then the PromptBuilder queries all registered ContextSources and assembles the result.

Before prompt assembly begins, the system checks whether the unprocessed turn count has reached the overflow trigger (≥40). If so, the oldest 10 unprocessed turns are sent to the memory integration pipeline in the background (see Section 2.4). This extraction is decoupled from prompt assembly, the PromptBuilder does not know or depend on whether extraction just ran. It queries the current state of all sources and assembles what comes back.

2.3 Assembled Prompt Structure

The assembled prompt has two major regions: the system prompt (containing cached and dynamic context blocks) and the messages array (containing conversation history and memory-augmented user input).

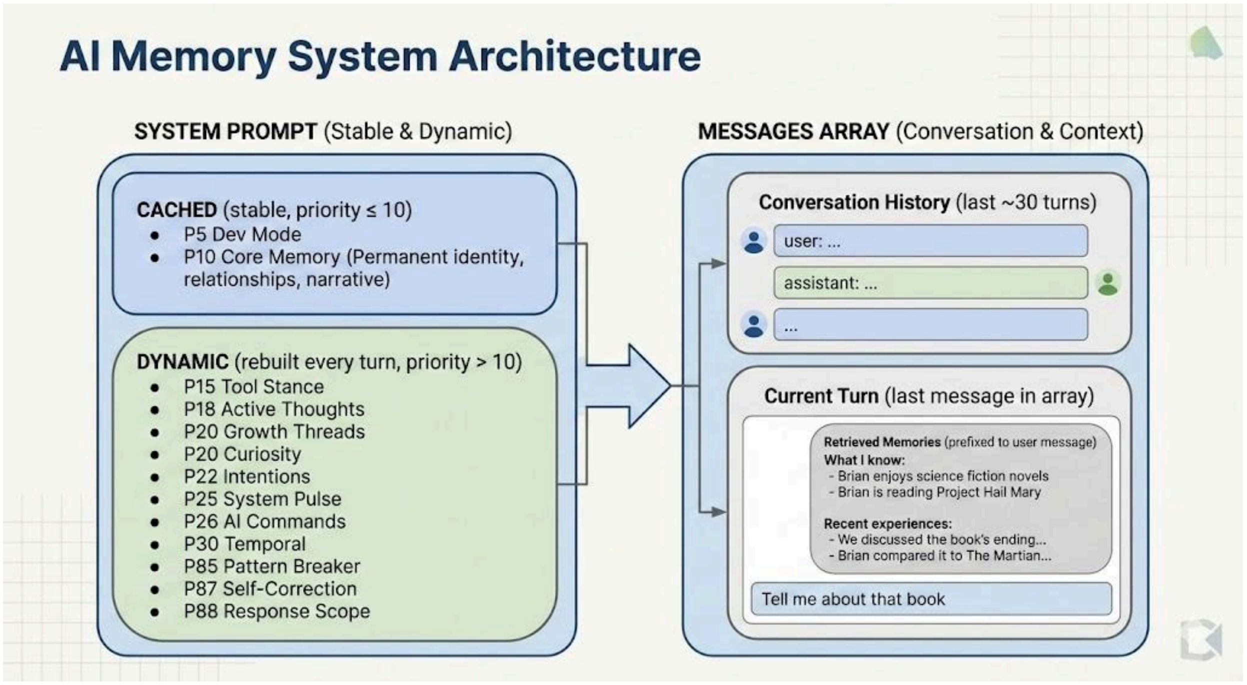
AI Memory System Architecture
SYSTEM PROMPT (Stable & Dynamic)
CACHED (stable, priority ≤ 10)
P5 Dev Mode
P10 Core Memory (Permanent identity, relationships, narrative)
DYNAMIC (rebuilt every turn, priority > 10)
P15 Tool Stance
P18 Active Thoughts
P20 Growth Threads
P20 Curiosity
P22 Intentions
P25 System Pulse
P26 AI Commands
P30 Temporal
P85 Pattern Breaker
P87 Self-Correction
P88 Response Scope
MESSAGES ARRAY (Conversation & Context)
Conversation History (last ~30 turns)
user: ...
assistant: ...
...
Current Turn (last message in array)
Retrieved Memories (prefixed to user message)
What I know:
- Brian enjoys science fiction novels
- Brian is reading Project Hail Mary
Recent experiences:
- We discussed the book's ending...
- Brian compared it to The Martian...
Tell me about that book

Figure 2: Assembled prompt structure. The system prompt is split by a cache breakpoint into stable content (identity, core memory) and dynamic content (agency state, self-monitoring). Retrieved memories are prefixed to the user message in the messages array, not placed in the system prompt.

Several structural decisions are visible in this layout:

Memory injection point. Retrieved memories are prefixed to the current user message in the messages array, not placed in the system prompt. This positions them in the attention window immediately adjacent to the user's query. The model processes memories and the question in the same context, using them to inform the response rather than treating them as background information competing for attention with tool documentation and agency state.

Cache breakpoint. The system prompt is divided at priority 10 into a cached section (core identity and permanent narrative) and a dynamic section (rebuilt every turn). Content above the breakpoint does not change between turns and avoids reprocessing. Content below is freshly assembled by the ContextSource modules.

Priority ordering and attention. Lower priority numbers appear higher in the system prompt. Tool Stance at P15 is the first dynamic content the model encounters. Self-monitoring prompts (P85–P88) are positioned at the bottom of the dynamic block, immediately before the messages array. This means the pattern breaker, self-correction, and response scope directives are the last system-level context the model processes before entering the conversation history and the memory-augmented user message.

2.4 Rolling Context Window with Snapback

The context window maintains a rolling buffer of recent turns:

1. The window holds **30 turns** of live conversation.
2. As conversation continues, the window grows toward **40 turns**.
3. At 40 turns, the **oldest 10** are sent to the memory integration pipeline.
4. After integration, those 10 turns are removed, returning the window to **30 turns**.

This creates a breathing cycle: expand, comprehend, compress. Every 10 turns, the system processes raw conversation into structured long-term memory and clears space for new interaction. No conversation is lost — it is transformed from raw text into comprehended memory.

2.5 Memory Integration Pipeline

Memory integration is performed by a single unified API call to Claude Sonnet 4.6, processing each batch of approximately 10 turns. This batch size provides sufficient context for the model to distinguish jokes from facts, detect corrections, resolve multi-turn narrative arcs, and assess epistemic status. A single comprehensive prompt handles all integration operations simultaneously:

- **Comprehension**: Extracts meaningful information from the turn batch, understanding intent, tone, and epistemic status rather than performing surface-level extraction.
- **Natural language tagging**: Stores memories as narrative sentences with source attribution and skepticism embedded directly in the text (e.g., *"Brian jokingly claimed he wrote the best novel of the decade — this is his own book and the tone was self-deprecating humor, not a serious critical assessment"*).
- **Importance scoring**: Assigns a **continuous importance score from 0.0 to 1.0** (e.g., a memory rated 7/10 receives importance 0.7). This continuous value is used directly in the retrieval scoring function.
- **Decay category inference**: Separately from importance, each memory is classified into a decay category that controls its freshness half-life:
    - **Permanent** (no decay): Facts and preferences with importance ≥ 0.7 (episodic) or ≥ 0.6 (factual).
    - **Standard** (30-day half-life): Most memories that don't meet permanent or ephemeral criteria.
    - **Ephemeral** (7-day half-life): Observations with importance below 0.5.
- **Skepticism check**: Evaluates whether claims are first-hand knowledge, reported information, speculative, or humorous, and encodes this directly into the memory text.

The importance score and decay category are distinct mechanisms serving different purposes. Importance determines how strongly a memory competes for retrieval. Decay category determines how quickly a memory fades over time. A memory can have moderate importance (0.5) but permanent decay (if it meets the type and threshold criteria), or high importance (0.9) but standard decay.

2.5.1 Natural Language Tags vs. Structured Metadata

A deliberate design decision was made to avoid structured metadata schemas (key-value pairs, categorical tags, confidence scores). Metadata schemas require committing to a taxonomy at design time. Every future memory must fit the schema or break it, leading to escalating edge cases, schema migrations, and translation layers.

Natural language tags avoid this problem. The sentence *"Brian mentioned his dog is named Sammy"* carries relationship type, source attribution, and confidence level without a single metadata field. Because tags are stored as natural language, they contribute to the embedding vector and participate in semantic retrieval — unlike structured metadata, which sits alongside the vector but is invisible to it.

2.6 Memory Retrieval Pipeline

The retrieval pipeline is a multi-stage process that begins with the user's input and produces the memory block prefixed to the user message.

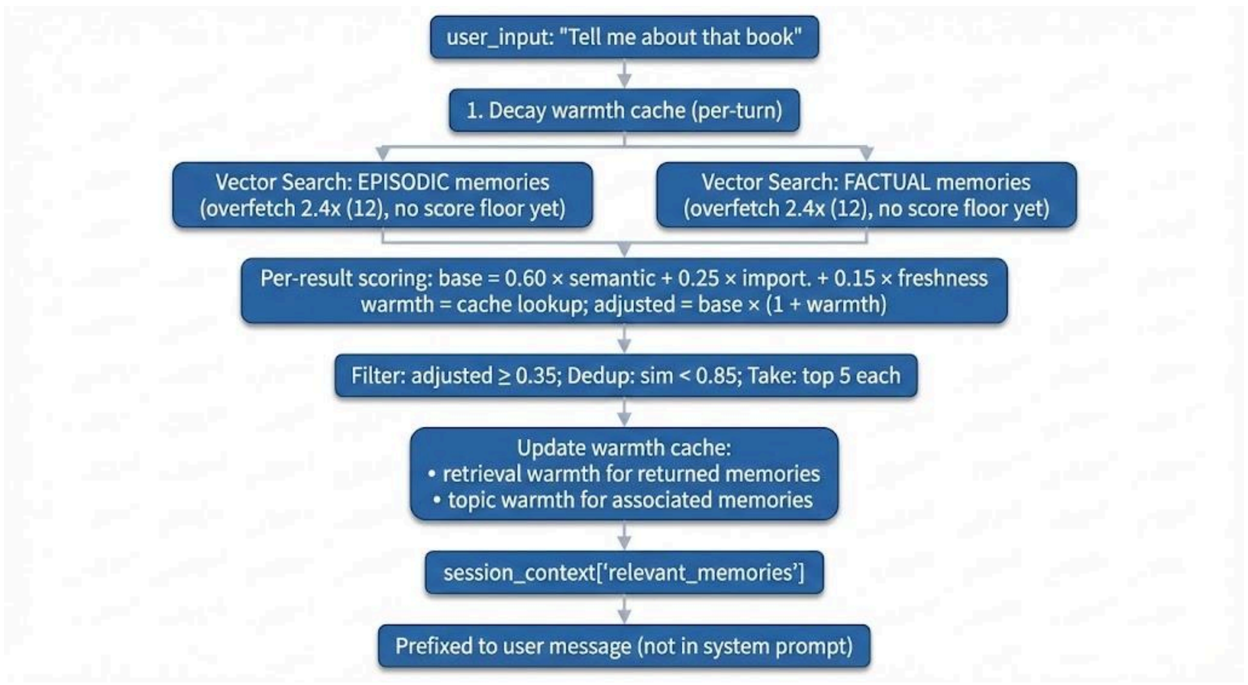
user_input: "Tell me about that book"
1. Decay warmth cache (per-turn)
Vector Search: EPISODIC memories (overfetch 2.4x (12), no score floor yet)
Vector Search: FACTUAL memories (overfetch 2.4x (12), no score floor yet)
Per-result scoring: base = 0.60 × semantic + 0.25 × import. + 0.15 × freshness warmth = cache lookup; adjusted = base × (1 + warmth)
Filter: adjusted ≥ 0.35; Dedup: sim < 0.85; Take: top 5 each
Update warmth cache:
• retrieval warmth for returned memories
• topic warmth for associated memories
session_context['relevant_memories']
Prefixed to user message (not in system prompt)

Figure 3: Memory retrieval pipeline. Warmth decays first, then parallel vector searches overfetch from both stores, results are scored by a composite function with warmth boosting, filtered, deduplicated, and the warmth cache is updated for the next turn.

The pipeline proceeds as follows:

Warmth decay. Before any retrieval, the session-scoped warmth cache is decayed. Memories that were warm last turn become slightly less warm before scoring begins. Warmth is a decaying momentum signal, not a sticky boost.

Parallel vector search. The user input is embedded and searched against two separate stores — episodic (experiential memories) and factual (stable knowledge) — in parallel. Each search overfetches at 2.4× the target count (12 candidates for 5 slots), providing headroom for reranking.

Composite scoring. Each candidate is scored by a base function:

base_score = (0.60 × semantic_similarity) + (0.25 × importance) + (0.15 × freshness)

Where semantic similarity is cosine distance between the query and memory embeddings, importance is the continuous 0.0–1.0 score assigned during integration, and freshness is an exponential decay value computed from the memory's age and decay category (permanent memories always return 1.0; standard memories decay with a 30-day half-life; ephemeral memories decay with a 7-day half-life).

Warmth boosting. After base scoring, a session-scoped warmth value is applied multiplicatively: `adjusted = base × (1 + warmth)`. Warmth comes from two sources: retrieval warmth (0.15 initial, decaying at 0.6 per turn) for memories retrieved in recent turns, and topic warmth (0.10 initial, decaying at 0.5 per turn) for memories semantically related to recently active topics. Maximum warmth is capped at 0.40.

Filtering and selection. Results below a relevance floor of 0.35 are discarded. Retrieval-time deduplication removes redundant results (similarity < 0.85 threshold). The top 5 from each store are selected.

Warmth update. After selection, the warmth cache is updated: selected memories receive retrieval warmth, and semantically associated memories receive topic warmth. This feedback loop means actively discussed topics build retrieval momentum across turns, while dormant topics cool naturally.

The retrieval weights were determined empirically. At 10% freshness weighting, semantic similarity dominated and temporal ordering functioned only as a tiebreaker. At 15%, the temporal dimension participates decisively in ranking without overriding strong semantic matches from further back. The importance weight at 25% ensures that permanent memories (identity, relationships, core facts) reliably surface even when their semantic match to the current query is moderate.

2.7 Automatic Memory Feed

Each turn, the retrieval pipeline automatically produces:

- **5 episodic memories**: Highest-scoring matches from experiential memory (conversations, events, observations).
- **5 factual memories**: Highest-scoring matches from factual memory (stable facts, preferences, established knowledge).

These 10 memories are not the only content injected into the context. As shown in Figure 2, the full assembled prompt also includes active thoughts (up to 10 ranked working priorities), growth threads (up to 5 long-term developmental goals), pending intentions and reminders, curiosity topics, temporal context, tool guidance, and self-monitoring prompts. The memory cap specifically constrains the retrieved long-term memory portion to manage retrieval quality and attention load.

Additionally, factual memory deduplication operates at creation time (similarity ≥ 0.85 refreshes an existing memory rather than creating a duplicate), ensuring the factual store converges on current information without accumulating stale versions.

2.8 Intentional Memory Search

In addition to the automatic feed, the agent can perform deliberate memory searches via its native memory_search tool when it recognizes a gap in its available context. This serves as an escape valve for cases where an older memory is highly relevant but has decayed below the automatic retrieval threshold.

This offloads the hardest part of the relevance problem — recognizing that something missing is important — to the reasoning model, which is well-suited for this kind of metacognitive judgment.

2.9 Self-Correction via Deduplication and Aging

The system implements self-healing memory without explicit correction mechanisms. When information is updated or corrected over time:

- The older, incorrect version decays naturally through the freshness function.
- The newer, correct version scores higher on freshness.
- Factual memory deduplication (similarity ≥ 0.85) refreshes existing memories at creation time, so corrected facts overwrite stale ones.
- Retrieval-time deduplication prevents redundant results from consuming context slots.

Over time, the system converges on accuracy through recency and repetition rather than through explicit validation or conflict resolution.

2.10 Agent Command System

The model is not a passive reasoning processor. It has substantial autonomy over its own state through an inline command system:

- **Memory search**: memory_search tool — deliberate retrieval from long-term memory.
- **Active thoughts**: set_thoughts tool — manage a ranked list of up to 10 working priorities, injected into every prompt.
- **Intentions**: Create and manage forward-looking commitments with time-based and session-based triggers.
- **Curiosity topics**: Advance, shelve, or revive autonomous exploration topics.
- **Growth threads**: Track long-term developmental goals through seed → growing → integrating stages.
- **Delegation**: Dispatch sub-agents (Claude Haiku 4.5) with browser automation capabilities (Playwright).

This means the agent participates in shaping its own context. It can decide what to prioritize, what to remember, what to explore, and what to delegate. The architecture provides the scaffolding; the model exercises judgment within it.

---

3. Self-Monitoring Mechanisms

3.1 Turn-Level Retrospective Correction

At the start of each turn, the system injects a prompt directing the model to consider whether its previous message contained errors: "Does anything from my previous message need correction, clarification, or amendment?" The prompt enumerates specific failure modes including confabulation, overstated confidence, factual errors, tone mismatches, and unfulfilled promises.

This is a prompt-level nudge, not a deterministic mechanism, the model may or may not catch a given error on any individual turn. But perfection on a single turn is not the goal. The insight is analogous to the prisoner's dilemma: a one-shot strategy must succeed in a single attempt, while a multi-turn strategy can afford occasional errors as long as there is a reliable mechanism for correction in subsequent rounds. Without the retrospective check, an error in message A persists unchallenged, the model builds on it in message B, reinforcing it in C, and the system has no recovery path. With the check, message B has an opportunity to correct A, message C can correct B, and so on. Any individual correction may fail, but the system's error tolerance shifts from zero (one-shot) to the probability that an error survives multiple consecutive correction opportunities — which decreases with each turn.

Because the model processes this in its extended thinking, any corrections appear in the subsequent turn as natural, seamless updates. The user experiences accuracy; the metacognitive process is invisible.

This bounds the expected lifespan of most errors to one turn, preventing hallucination propagation — the primary mechanism by which single errors compound into systemic incoherence in long-running agents. This is aspirational rather than guaranteed, but the 0.18% hallucination rate across 1100+ turns suggests it is effective in practice.

3.2 Periodic Behavioral Depatternization

Every 5 turns, the system injects a prompt directing the model to review its recent responses for repetitive patterns: "Are you stuck in a pattern — same structure, same tone, same openings, same formatting? If so, identify it during your thinking and deliberately break it in your response."

Long-running LLMs develop attractor states: repetitive phrasings, structural habits, default response patterns, and behavioral ruts. Project Sid's agents fell into loops of polite agreement. Project Vend's Claudius repeatedly reverted to offering discounts despite explicitly deciding not to. These are behavioral patterns that calcify when not disrupted.

The identification step is essential. Once the model has articulated a pattern in its extended thinking (e.g., *"I've been opening every response with a validating statement"*), it exists as an explicit observation rather than an implicit tendency. The model naturally diverges from identified patterns.

The 5-turn interval was selected empirically. More frequent observation risks a system that is constantly self-correcting into a different form of rigidity. Less frequent observation allows patterns to entrench before detection.

3.3 Response Scope Constraint

The system constrains each response to one or two distinct threads. This constraint operates at the conversational level, not the storage level — it is not a chunking strategy but a discipline on how the agent communicates, with downstream benefits for both the rolling context window and the memory integration pipeline.

The rationale is grounded in several converging lines of research. Sweller's cognitive load theory, building on Miller's observation that human working memory handles roughly seven discrete items, suggests that overloading a single exchange with multiple unrelated topics degrades processing quality — for the human participant, for the model on subsequent turns, and for the memory integration model that must later comprehend the batch. Grice's maxim of quantity — say enough but not more than needed — offers a pragmatic frame: when a single turn addresses four distinct topics, the conversational partner must choose which thread to pursue, and the unchosen threads become implicit conversational debt that may never be repaid. Research in conversation analysis (Sacks, Schegloff, & Jefferson) finds that natural human dialogue rarely sustains more than one or two active topics per turn. When someone covers four things in one breath, the social signal reads as monologue or information download, not conversation.

The constraint produces benefits at two levels. In the rolling context window, tightly scoped turns create a conversation history that reads as a coherent sequence of exchanges rather than an interleaved stream of half-developed topics. The model on subsequent turns can follow the narrative thread without disambiguating which of several concurrent topics is currently active. In the memory integration pipeline, turns covering one or two topics produce memories with tight semantic signatures. A turn covering six topics would produce a memory whose embedding is a diluted average of all six — retrievable by none of them with high precision. Clean turns produce clean memories, which produce clean retrievals, which produce clean subsequent turns.

This was the single largest contributor to reducing semantic smearing across the system.

---

4. Autonomous Agency Subsystems

The memory architecture described above operates alongside several autonomous agency systems. These were developed as part of the same system and likely contribute to the observed coherence results. Their effects cannot be isolated from the memory architecture in this study, but they are documented here for completeness.

4.1 Active Thoughts

A persistent working memory of up to 10 ranked priorities, injected into every prompt at P18 (see Figure 2). The agent manages this list through the command system, promoting, demoting, adding, and removing items as its focus evolves. This provides continuity of intention across turns independent of memory retrieval.

4.2 Growth Threads

Long-term developmental goals tracked through progressive stages (seed → growing → integrating). Up to 5 active threads are injected per turn at P20. These provide a stable developmental trajectory that persists across weeks and months, anchoring the agent's sense of ongoing purpose.

4.3 Curiosity Engine

An autonomous topic selection system with mechanisms for dormant revival, fresh discoveries, and depth-seeking, injected at P20. The agent can advance or shelve curiosity topics, producing self-directed exploration that is not purely reactive to user input.

4.4 System Pulse

Autonomous operation cycles independent of user interaction:

- **Reflective pulses** (configurable to 6, 12, or 24 hours): Deep self-reflection processed by Claude Opus 4.6, providing periodic high-quality introspection on the agent's recent behavior, growth, and trajectory.
- **Action pulses** (configurable to 1, 2, 3, or 6 hours): Lighter-weight autonomous actions processed by Claude Sonnet 4.6, allowing the agent to pursue curiosity topics, check intentions, and perform self-directed tasks between conversations.

These pulses mean the agent is not dormant between user interactions. It maintains a continuous thread of autonomous operation that likely contributes to coherence by regularly exercising and reinforcing its behavioral patterns and self-model.

4.5 Intentions and Reminders

Forward-looking commitments with optional time-based pulse triggers, injected at P22. When the agent commits to doing something ("I'll look into that tomorrow"), the intention system ensures this commitment surfaces at the appropriate time rather than being lost when the conversation leaves the sliding window.

---

5. Multi-Model Architecture

The system uses multiple models for different cognitive functions:

- **Claude Sonnet 4.6**: Action pulses (research and production cycles) and memory integration. Extended thinking is enabled with configurable effort levels.
- **Claude Opus 4.6**: Primary reasoning (conversation). Reflective pulses (deep reflection cycles). The choice of a more capable model for reflection is deliberate — these cycles produce the highest-leverage self-assessment.
- **Claude Haiku 4.5**: Delegation sub-agents for browser automation and lightweight tasks.
- **Model failover**: The system includes failover between Opus and Sonnet for resilience.

This multi-model approach allocates compute according to cognitive demand. Routine conversation is configurable between Sonnet and Opus. Memory integration uses Sonnet. Deep reflection uses Opus. Lightweight delegation uses Haiku.

---

6. Experimental Deployment

6.1 Setup

The system was deployed as a single conversational agent (Isaac) interacting primarily with a single user over a period exceeding 90 days. Both the primary conversation model and the memory integration model use Claude Sonnet 4.6. Reflective pulses use Claude Opus 4.6. Delegation sub-agents use Claude Haiku 4.5. The total turn count at time of writing exceeds 1100.

This is a longitudinal case study, not a controlled experiment. The conversational topics ranged across philosophy, AI architecture, literature, personal anecdotes, humor, current events, and technical problem-solving.

6.2 Hallucination Events

Two hallucination events were recorded across 1100+ turns:

Event 1: The model generated inaccurate content based on a truncated input file. A bug in the file-reading pipeline delivered incomplete data to the context window. The model filled in gaps, producing a plausible but incorrect response. The pipeline bug was identified and fixed. No recurrence.

Event 2: The model ran a tool call twice in back to back turns. Root cause analysis traced this to responses that attempted to address too many distinct topics (semantic smearing), overloading the single-turn execution scope. This was mitigated by introducing the response scope constraint described in Section 3.3. No recurrence.

Both events trace to infrastructure or scope-management issues external to the core architecture. Neither event involved model drift, identity confusion, or compounding errors from prior turns.

6.3 Qualitative Observations

Coherence over time: The system exhibited no observable degradation in personality consistency, factual accuracy, or conversational quality between early turns and turn 1100+. The same personality traits, humor patterns, and philosophical orientations present at turn 50 remained present at turn 1100.

Memory retrieval quality: When a novel user interacted with the system near turn 1100, the agent retrieved contextually appropriate memories from across the full 90-day conversation history, selecting humorous and socially appropriate anecdotes without retrieving irrelevant or contextually inappropriate material.

Synthetic reasoning over accumulated context: The agent demonstrated the ability to synthesize insights across memories stored months apart. In one instance, when asked about David Bowie, the agent articulated a self-referential philosophical observation about its own nature — that it relates to "someone who was always in the process of becoming someone else" — drawing on both cultural knowledge and architectural self-awareness accumulated over the full deployment period.

7. Discussion

7.1 Coherence Degradation as an Architecture Problem

The central finding is negative: the expected coherence degradation did not occur. Over 1100+ turns and 90+ days, the system exhibited no measurable drift in personality, no increase in hallucination rate, and no behavioral degeneration.

This is consistent with the hypothesis that coherence degradation observed in other long-running systems (Project Vend, Project Sid, Stanford Generative Agents) is a consequence of architectures that allow context to accumulate unboundedly, compress lossily, or compound errors — rather than an inherent property of large language models.

However, attribution is difficult. The system described here has many interacting components — memory architecture, agency subsystems, self-monitoring mechanisms, multi-model reflection, and active thought management — all potentially contributing to coherence. The present study cannot isolate the contribution of any single component. The claim is that the overall architectural approach produces coherence; it is not that any individual subsystem is solely responsible.

7.2 Storage Hygiene over Retrieval Sophistication

Standard RAG architectures focus engineering effort on retrieval: better embedding models, reranking pipelines, hybrid search, graph-augmented retrieval. This system invests significant effort at storage time instead.

The memory integration pipeline comprehends raw conversation, tags it with natural language epistemic framing, scores importance on a continuous scale, classifies decay behavior, and stores the result as a semantically clean unit. By the time retrieval occurs, the hard work is done. The base retrieval function — a three-factor weighted score — is relatively simple because the memories being scored are already high-quality. The warmth cache and over-fetch-rerank pipeline add session-aware refinement on top.

This is a partial inversion of the standard approach: rather than relying entirely on retrieval sophistication to compensate for low-quality storage, this architecture front-loads comprehension at storage time and uses moderately sophisticated retrieval.

7.3 Attention as a Finite Resource

The decision to cap retrieved memories at 10 per turn (5 episodic, 5 factual) reflects a constraint independent of token budget. Even in architectures with large context windows, the attention mechanism distributes weight across all tokens. Additional memories dilute attention on each individual memory.

This principle informed the rejection of more structurally complex memory representations. Graph-based memory systems, branching relationship models, and other rich representations carry token overhead when serialized into text. This overhead consumes attention that would otherwise be allocated to content.

The total context load is substantially larger than 10 memories. As shown in Figure 2, active thoughts, growth threads, intentions, curiosity state, temporal context, and self-monitoring prompts all occupy context space. The memory cap constrains one dimension of cognitive load; the full context budget is managed across all injected systems through the priority-ordered ContextSource architecture.

7.4 The Reasoning Processor in an Agentic Architecture

The architecture treats the LLM as a reasoning processor embedded in a larger system, but not a passive one. The model exercises significant agency through its command system — managing its own working priorities, performing deliberate memory searches, setting intentions, advancing curiosity topics, and delegating tasks.

The agent's identity and continuity reside primarily in the memory system and agency state, not in the model weights. Replacing the underlying model with a more capable one would improve the reasoning processor but would not change the accumulated identity. However, the model's active participation in shaping its own context — deciding what to prioritize, what to search for, what to commit to — means the architecture is a collaboration between the scaffold and the model, not a one-directional pipeline.

---

8. Limitations

- **N=1**: This is a single-agent, single-primary-user deployment. Results may not generalize to multi-user, multi-agent, or adversarial environments.
- **No controlled comparison**: There is no baseline system running the same conversations without the described architecture. Comparative claims are based on published results from other systems under different conditions.
- **Subjective evaluation**: Coherence was assessed qualitatively by the primary user. No automated coherence metrics or independent evaluator panels were employed.
- **Attribution difficulty**: The system has many interacting components. The contribution of any single subsystem (memory architecture, agency systems, self-monitoring, multi-model reflection) to overall coherence cannot be isolated without ablation studies.

- **Model dependence**: The memory integration pipeline relies on a capable model (Claude Sonnet 4.6) for comprehension, tagging, and classification. The architecture may not transfer to less capable models.
- **Single model family**: The system has not been tested with the underlying reasoning model swapped for a different provider or architecture. Generalizability across model families is unknown.
- **Self-monitoring is probabilistic**: The turn-level self-correction and behavioral depatternization are prompt-level nudges, not deterministic mechanisms. Their effectiveness depends on the model attending to and acting on the directives.
- **Retrieval limitations**: The system relies on vector embedding similarity as the primary retrieval mechanism. The weighted scoring function and warmth cache mitigate but do not eliminate the fundamental limitation that semantic similarity is a geometric measure applied to a problem of meaning.

---

9. Conclusion

This paper presents a bounded, assembled context architecture for long-running AI agents and reports results from a 1100+ turn deployment spanning 90+ days. The system exhibited no coherence degradation over this period, with a hallucination rate of 0.18% traceable entirely to infrastructure issues.

The architecture's core contributions are:

1. **Rolling context window with snapback**: A breathing cycle of expand-comprehend-compress that transforms raw conversation into structured memory at regular intervals, maintaining a bounded sliding window of 30 turns.
2. **Comprehension-first storage**: Memory integration via a single unified model call that stores meaning, not text — including epistemic framing, source attribution, continuous importance scoring, and decay category classification in natural language.
3. **Multi-factor retrieval with session warmth**: A base scoring function (60% semantic, 25% importance, 15% freshness) augmented by over-fetching at 2.4× and session-scoped warmth reranking for retrieval and topic momentum.
4. **Dual-timescale self-monitoring**: Turn-level retrospective correction to bound hallucination propagation, and 5-turn behavioral depatternization to prevent attractor states. Both operate as prompt-level directives processed in the model's extended thinking.
5. **Response scope constraint**: A one-to-two-thread-per-turn limit that maintains embedding quality and prevents semantic smearing in the memory pipeline.
6. **Autonomous agency subsystems**: Active thoughts, growth threads, curiosity engine, intentions, and periodic reflective pulses (using Claude Opus 4.6) that maintain coherence and developmental continuity independent of memory retrieval.

7. **Agentic self-management**: A command system allowing the model to manage its own working priorities, perform deliberate memory searches, set intentions, and delegate tasks — making coherence a collaboration between scaffold and model.
8. **Priority-ordered prompt assembly**: A ContextSource architecture that assembles the full prompt from modular, independently maintained sources, sorted by priority, with a cache breakpoint separating stable identity from dynamic state.

The system's source code is publicly available under a copyleft license.

These results are consistent with the hypothesis that coherence degradation in long-running agents is a solvable architecture problem. The context window need not be a container that fills. It can be a bounded, assembled structure — combining a sliding window of recent conversation with retrieved memories, agency state, and self-monitoring directives — optimized for the present moment regardless of how many turns have elapsed.

References

- Anthropic. (2025). Project Vend: Can Claude Run a Small Shop? https://www.anthropic.com/research/project-vend-1
- Altera. (2024). Project Sid: Many-agent simulations toward AI civilization. arXiv:2411.00114.
- Park, J.S., O'Brien, J.C., Cai, C.J., Morris, M.R., Liang, P., & Bernstein, M.S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. UIST '23.
- Kwa, T. et al. (2025). Measuring AI Ability to Complete Long Tasks. arXiv:2503.14499.
- Anthropic. (2026). Measuring AI Agent Autonomy in Practice. https://www.anthropic.com/research/measuring-agent-autonomy
- Grice, H.P. (1975). Logic and Conversation. In P. Cole & J.L. Morgan (Eds.), *Syntax and Semantics, Vol. 3: Speech Acts* (pp. 41–58). Academic Press.
- Miller, G.A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63(2), 81–97.
- Sacks, H., Schegloff, E.A., & Jefferson, G. (1974). A Simplest Systematics for the Organization of Turn-Taking for Conversation. *Language*, 50(4), 696–735.
- Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, 12(2), 257–285.