A Cognitive Architecture for AI Agent Memory

Modeling Artificial Memory Systems on the Human Multi-Store Model

A Technical Whitepaper — Version 1.1 (Revised) Originally published May 2026 · Revised May 2026

By Kelsi Guidry *with research and drafting assistance from Claude (Anthropic)*

Abstract

Most production AI agents today operate without genuine memory. They reason brilliantly within a single conversation, then begin the next one as strangers. The dominant remediation pattern — bolting a vector database to a large language model and calling the result Retrieval-Augmented Generation — addresses the symptom but not the underlying architecture. Memory is not a database lookup. In the only system known to do memory well, the human brain, memory is a *layered process* with distinct stores, distinct dynamics, and an active consolidation step that decides what is worth keeping.

This paper proposes a cognitive architecture for AI agents modeled on the Atkinson–Shiffrin multi-store model of human memory, extended with the modern understanding of memory consolidation, retrieval cues, and use-dependent reinforcement. We describe five functional layers — sensory ingestion, short-term working memory, the consolidation bridge, long-term memory in three co-located stores, and retrieval as an active cognitive operation — and discuss the engineering implications of each. The aim is not to claim that AI agents must mimic biology in detail, but to argue that the *functional decomposition* the human brain has settled on is a better blueprint than the flat retrieval pipelines that currently dominate the field.

---

1. The Memory Problem in Modern AI Agents

A large language model, in isolation, has no persistent memory. Whatever it appears to "know" was either embedded in its weights at training time — frozen, generic, and unchangeable per user — or pasted into its context window for the duration of a single inference. Once that inference returns, the model forgets everything that just happened.

This is a strange limitation for systems we increasingly trust with agentic work. A human assistant who could reason at the level of a frontier LLM but woke up each morning with no memory of the previous day would be unemployable. Yet this is the default condition of an LLM agent, and the patches the industry has applied to it have largely been external.

The most common patch is Retrieval-Augmented Generation: maintain an external store of documents, embed each as a vector, and at inference time retrieve the top-k semantically similar documents and inject them into the model's context. RAG is useful, and we will return to it as one component of a larger system. But RAG alone treats memory as *storage and lookup*. It collapses what cognitive psychology recognizes as at least four distinct functions — encoding, consolidation, storage, and retrieval — into a single operation. It has no notion of importance, no notion of recency, no notion of decay, no notion of why this memory exists or what it supersedes. It is a filing cabinet attached to a brilliant amnesiac.

Du's March 2026 survey of LLM agent memory closes with what may be the cleanest statement of the open problems in the field: the hardest questions remaining are "how to consolidate without catastrophic loss, how to retrieve by cause rather than similarity, how to reflect without entrenching errors, and how to forget safely. Solving these will determine whether the next generation of agents is merely impressive or genuinely reliable." [Du, 2026] The same paper observes that "memory deserves the same level of engineering investment as the LLM itself. Model selection gets months of careful benchmarking; memory architecture often gets an afternoon." This whitepaper takes that diagnosis as its starting point and argues that the human brain offers a remarkably good blueprint for getting memory architecture right.

---

2. What the Brain Actually Does

To build a memory system worth modeling, we should first understand what we are modeling. Cognitive psychology has converged on a layered account of human memory that is more than half a century old and remains broadly accepted, with refinements.

The foundational framework is the multi-store model proposed by Atkinson and Shiffrin in 1968. The model asserts that human memory has three separate components: a sensory register, where sensory information enters memory; a short-term store; and a long-term store. Information flows through these stores under the control of attention, rehearsal, and retrieval — and at each transition, most of what came before is discarded.

2.1 Sensory memory

The sensory register is the first stage. It has unlimited capacity but a duration of roughly 250 milliseconds to 2 seconds before it fades. Every sight, sound, taste, and touch enters here. Images are stored visually as iconic memory; sounds are stored as echoic memory. The store

has a large capacity but a very brief duration; most of the information is lost through decay before higher cognitive processes can act on it.

The sensory register's role is to act as a buffer — to hold the raw signal long enough for higher cognitive processes to decide whether any of it matters. Atkinson and Shiffrin called these registers "buffers" specifically because their function is to prevent the immense flow of incoming sensory data from overwhelming higher-level processing. Transfer from sensory memory to the next stage is gated by attention. Without attention, the information decays and is lost.

2.2 Short-term (working) memory

If a sensory impression is attended to, it moves into short-term memory. Short-term memory is famously small — Miller's "magical number seven, plus or minus two" — and famously brief, with items decaying within roughly 15 to 30 seconds without active rehearsal.

The modern refinement of this idea is *working memory*, articulated most influentially by Baddeley [Baddeley, 2000]. Working memory is not merely passive storage; it is the active workspace where reasoning happens. New input from the senses combines with retrieved information from long-term memory in this workspace, and conscious thought is the activity that takes place there.

2.3 Long-term memory

If working memory content is rehearsed or made meaningful, it can be transferred to long-term memory, which has effectively unlimited capacity and durations measured in years to decades. The transfer is not automatic — it requires *encoding*, a process that gives information meaning by connecting it to what is already known.

Long-term memory is itself differentiated. The standard division separates *declarative* memory (facts and events one can consciously articulate) from *non-declarative* or *procedural* memory (skills and habits). Declarative memory further splits into *episodic* memory (specific personal experiences, anchored in time and place) and *semantic* memory (general knowledge about the world, abstracted from any particular episode) [Tulving, 1972; Squire, 2004]. These distinctions matter for AI architecture because they imply that a single flat store cannot adequately serve all retrieval needs.

2.4 Consolidation

Memory consolidation is the process that moves information from short-term to long-term storage. It is not instantaneous, and it is not lossless. Modern neuroscience has shown that the hippocampus initially binds new episodic memories, and over hours to days these traces are gradually integrated into neocortical storage — a process that depends critically on sleep, particularly the slow-wave and REM phases. During consolidation, memories are not just

moved; they are reorganized, abstracted, connected to existing knowledge, and selectively pruned.

This is the most important and most overlooked stage when translating brain function into AI architecture. Consolidation is *active cognitive work*. It decides what is worth keeping, how to integrate the new with the old, what should supersede what, and what can be forgotten.

2.5 Retrieval

Retrieval is the process of pulling a memory back into working memory so it can be used. It is not lookup; it is reconstruction. Memories are stored in associative webs, and retrieval typically proceeds by *cue* — a partial match that activates connected memories. The smell of a place, a fragment of a melody, a name that resembles another name: any of these can serve as a cue that surfaces a richer memory by association.

Retrieval also has a feedback effect. Each successful retrieval strengthens the memory and the path to it. This is the *testing effect*, well-established in educational psychology: actively recalling something reinforces it more than passive review. Conversely, memories that are never retrieved decay — not because they are deleted, but because the path to them weakens until they become effectively unfindable.

This suggests that memory is not static storage. It is dynamic, use-shaped, and partially constructive. Any AI architecture that ignores these dynamics will exhibit pathologies that AI memory systems already exhibit: stale information persisting forever, frequently-needed information being just as hard to find as a one-off note, and no mechanism for one memory to *replace* another when reality has changed.

---

3. The Five-Layer Cognitive Architecture

We propose a five-layer architecture that maps each stage of human memory onto a concrete engineering primitive. The layers are: sensory ingestion, short-term memory, the consolidation bridge, long-term memory, and retrieval. Each layer has a clear functional contract with the layers above and below it, and each can be implemented with off-the-shelf technology.

This decomposition is consistent with the broader trend in cognitive architectures for language agents. Sumers et al.'s *Cognitive Architectures for Language Agents* propose a generalized blueprint where working, episodic, semantic, and procedural stores interact through a central executive (the LLM) [Sumers et al., 2024]. The framework presented here is a more operational refinement, organized around the *flow of information through stages* rather than the *types of stored content*, on the principle that the flow is the engineering challenge.

3.1 Layer 1: Sensory ingestion

The sensory layer is the agent's intake surface. Every input the agent receives — user messages, voice transcripts, uploaded files, fetched URLs, tool outputs, observations from other agents — lands here first, in raw form, with provenance metadata attached.

The functional contract of this layer mirrors the human sensory register: capture everything, hold it cheaply, do not yet attempt to understand it. The decision about what is worth keeping happens later. The sensory layer's only job is to ensure that no signal is lost between the moment it arrives and the moment a higher process decides whether it matters.

In engineering terms, this is an ingestion API and a raw object store. Inputs are normalized into a common envelope — source, timestamp, modality, agent context, raw content — and written to durable storage indexed by a stable identifier. They are not yet embedded, summarized, or interpreted. Sensory storage is large, cheap, and append-only.

The discipline this layer enforces is *provenance*. Every memory in the system, no matter how processed and abstracted it eventually becomes, must trace back to a sensory record. When an agent later recalls a fact, the system can answer the question "where did I learn this?" — a question that LLMs without provenance cannot answer truthfully and often answer with hallucinated sources. In a B2B or otherwise accountable deployment, this is not optional.

3.2 Layer 2: Short-term memory

Short-term memory is the agent's active workspace — the analog of a session, a conversation, a current task. It is where new sensory input is combined with retrieved long-term memory to produce the next action or response.

The defining property of short-term memory is *bounded capacity*. The LLM's context window is the hard ceiling, and within that ceiling we must fit a working set: the recent dialog, currently relevant retrieved memories, the active task state, and the room the model needs for its own reasoning tokens. Pushing past this ceiling does not produce a graceful degradation; it produces silent forgetting of the earliest content. As a recent survey notes, "long context is not memory" — long-context models consistently underperform purpose-built memory systems on tasks requiring selective retrieval and active management [Du, 2026].

Engineering-wise, the short-term layer combines a fast in-memory cache (for the active session's dialog and working set) with a structured session record (for what the session contained, when it ran, and what entities it touched). Sessions have explicit boundaries — they open, they accumulate turns, and they close — and the closing event is what triggers the next layer.

A useful architectural commitment at this layer is to treat the LLM's context window as *managed* rather than as the user's responsibility. The agent should know its own working-memory budget, decide what to keep in the active context vs. what to push out to retrieval-on-demand, and explicitly track what is currently "in mind" vs. what is available but not loaded. This corresponds to the human distinction between what one is *currently thinking about* and what one *could think about with a moment's effort*.

3.3 Layer 3: The consolidation bridge

The consolidation bridge is the most architecturally important layer in this framework, and the layer most often missing from contemporary agent designs.

Consolidation is the active cognitive process that turns the residue of a closed session into long-term memory. It is not summarization, although summarization is part of it. It is not embedding, although embedding is part of it. It is the process that asks, of every closed session: *what here is worth keeping, how should it be encoded, what does it supersede, and what should be discarded?*

The engineering parallel is an asynchronous worker that runs after a session closes (and on a periodic schedule for orphaned input that did not belong to any session). The worker takes the session transcript, the linked sensory records, and a summary of what is already known, and applies a reasoning model to produce structured output: a set of candidate memories, each with a title, a summary, a body, semantic tags, extracted entities, a relevance score, and an indication of whether the new memory supersedes any existing one.

This pattern is beginning to emerge across production-grade agent frameworks, where a vocabulary shift is underway: memory is increasingly treated as cognition rather than storage, with each memory operation — encode, consolidate, recall, extract, forget — implemented as an active reasoning process powered by LLM analysis. The cost of this approach is real: encoding analysis adds one to two LLM calls per write, with corresponding latency. But the cost is paid asynchronously, off the critical path of user interaction, and is the price of having an agent that actually develops useful long-term knowledge rather than accumulating noise.

Three responsibilities make this layer non-trivial and worth treating as a first-class subsystem:

Relevance scoring. Not every closed session deserves to leave a permanent trace, and not every part of a kept session deserves equal weight. The consolidator must score candidate memories on a continuum from "explicit decisions and named commitments" (high relevance) to "ambient context that may be useful by association" (medium) to "small talk and exact duplicates" (discard). The threshold is configurable per agent and per tenant; some agents should remember more, others less.

Supersession. When a new memory contradicts or updates an old one, the system must record the relationship explicitly rather than letting both coexist. If on Monday the agent learned that the customer uses PostgreSQL, and on Friday it learned that the customer migrated to MySQL, the correct outcome is not two memories with different facts. The correct outcome is a Friday memory that supersedes the Monday memory, with the Monday memory preserved but flagged as historical. This is essential for any agent that operates over real time horizons, and it directly addresses what the literature flags as "staleness, contradictions, and drift" — failure modes that long-lived memory systems exhibit when supersession is not made explicit [Du, 2026].

Encoding. The consolidator decides the *form* in which a memory will be stored. A raw transcript is not a memory; it is the source material from which a memory is encoded. The encoded memory is a self-contained, human-readable artifact — typically a structured text document with frontmatter metadata — that says what it knows, why, and from where. Encoding for meaning rather than verbatim storage is the AI analog of the cognitive principle that depth of processing predicts retention.

The biological metaphor for this layer is sleep. Just as the human brain consolidates the day's experiences during slow-wave and REM cycles, the consolidation worker runs on a schedule decoupled from real-time interaction. This is not poetic; it is the right engineering pattern. Doing consolidation inline with user interaction couples response latency to LLM throughput on a heavyweight task, and doing it as a fragile cron loses retries, observability, and the ability to replay. A proper queue-based worker — with retries, dead-letter handling, and explicit job state — is the correct primitive.

3.4 Layer 4: Long-term memory

Long-term memory is where consolidated memories live. The proposal here is that long-term memory should not be a single store but a *coordinated set of three co-located stores*, each playing a distinct role.

The canonical store: human-readable encoded memories. Every memory exists primarily as a structured text document — markdown with frontmatter is the natural choice — written to a persistent, version-controllable file system. Each file contains the encoded memory in prose along with metadata: identifiers, tenant scope, agent of origin, tags, extracted entities, relevance score, source provenance, and timestamps. This is the *source of truth* for the memory's content. Everything else in the long-term layer is derived from it.

The reason for this commitment is durability and operability. A markdown file can be read by a human, edited by hand if the consolidator made a poor judgment, version-controlled to track how a memory evolved, grep'd from the command line, and migrated between systems without loss. If the index built on top of these files is corrupted or replaced, the memories themselves survive untouched.

The retrieval index: vectors and structured metadata. A vector database — pgvector, Qdrant, LanceDB, or similar — holds embeddings of each memory's title, summary, and body, alongside structured metadata that mirrors the markdown frontmatter. This is the *query surface* of long-term memory. It is fast, it supports semantic similarity and filtered retrieval, and it can be rebuilt from the canonical store at any time.

The architectural insight is that the canonical store and the retrieval index are different kinds of object. The canonical store is optimized for writing once and being trusted forever. The index is optimized for being queried millions of times and being rebuildable when the embedding model changes (which it will). Conflating them — putting the only copy of memory content inside a vector database — is a design choice many teams come to regret on the day they want to switch embedding providers.

The provenance archive: raw sensory records. The original raw inputs that fed each memory remain in the sensory archive, with stable references back from each consolidated memory. This is queried rarely — typically only when a user asks "where did you learn this?" or when a memory needs to be reconsolidated from source. But its existence is what allows the agent to be *truthful about its own history*, which matters more in production deployments than is often appreciated.

This three-store decomposition mirrors a pattern that has emerged in mature data architectures generally: the separation of system of record (canonical), system of query (index), and system of provenance (archive). Memory systems benefit from the same separation for the same reasons.

The four cognitive memory types — working, episodic, semantic, and procedural — do not require four physical stores in this architecture. They emerge as *views* over the same underlying memories, distinguished by metadata: episodic memories are those tagged as specific events with temporal anchors; semantic memories are those that have been abstracted from one or many episodes into general statements; procedural memories are those tagged as workflows or learned skills. Working memory is the short-term layer.

3.5 Layer 5: Retrieval as active cognition

Retrieval, in this framework, is not a database lookup. It is an active cognitive operation that happens on every agent turn and that uses multiple strategies in combination.

The simplest strategy is *semantic retrieval*: embed the current query, find the top-k memories by vector similarity, return them. This is RAG in the conventional sense, and it remains useful as a baseline. But it has well-known failure modes — it ranks by surface similarity rather than relevance, it has no notion of recency, and it has no notion of which memories the agent has actually found useful in the past.

A cognitively-grounded retrieval layer combines semantic retrieval with two additional dynamics:

Recency and decay. Recent memories are slightly preferred over old ones, all else equal. This is not a hard filter — old memories must remain retrievable when relevant — but a soft boost that reflects the cognitive reality that recent context is more often what matters. The decay function is gentle and configurable. *Decay here means deprioritization in ranking, never deletion.*

Use-dependent reinforcement. Every successful retrieval bumps a memory's retrieval weight, and that weight feeds back into future ranking. Memories that are frequently retrieved become easier to retrieve; memories that are never retrieved sink to the bottom of the ranking but remain in storage. This is the engineering analog of the testing effect and the "use it or lose it" principle in human memory. Crucially, it is not deletion — the memory remains, in case the cue that activates it eventually arrives — it is just deprioritization. The Generative Agents work [Park et al., 2023] introduced a related multi-signal scoring mix of recency, relevance, and self-assessed importance that has remained influential; a vector-based activation mechanism incorporating temporal decay, semantic similarity, and probabilistic noise — drawn from the ACT-R cognitive architecture [Anderson, 1983] — provides one mathematical formulation of the same idea.

A complete retrieval layer also exposes more than one *mode* of retrieval, because human memory itself has more than one. Beyond semantic similarity, an agent benefits from:

Recall-by-filter — exact metadata queries for when the agent already knows what it is looking for. "Get me every memory tagged with this client" is not a similarity question; it is a structured query. Forcing it through vector search degrades both speed and precision.

Cue-based bridging — given a starting memory, return memories that share entities, tags, or temporal proximity. This corresponds to the human experience of one memory cueing another by association, and it is the right pattern when the agent is exploring rather than answering.

In practice, the retrieval layer should be invoked on every agent turn with a strategy chosen by the agent itself: semantic when answering a free-form question, recall when working with a known scope, bridge when reflecting or planning. The output is a ranked set of memories, each loaded from the canonical store and assembled into the working memory context.

---

4. Cross-Cutting Properties

Three properties run through all five layers and are worth surfacing on their own.

4.1 Tenant isolation as a primitive

If the agent serves more than one user, organization, or context, every record at every layer must be tenant-scoped from the schema up. This is not a feature to be added later; it is a property of the data model. Sensory records, sessions, consolidated memories, embeddings, and raw archives all carry a tenant identifier, and every retrieval is filtered by it. The cost of designing this in from the beginning is small. The cost of refactoring it in later is large and accident-prone.

4.2 Provenance as a first-class output

Every consolidated memory links to its source sensory records. Every retrieval returns not just the memory content but the chain back to where it came from. This enables the agent to answer "how do you know that?" with a real answer rather than a fabricated one. In agentic deployments where the agent's outputs feed downstream decisions — financial, medical, legal, operational — this property is the difference between a tool that can be trusted and one that cannot.

4.3 The forgetting question

Forgetting is uncomfortable to design and uncomfortable to discuss, but it is part of any honest memory architecture. Three forms of forgetting deserve explicit handling:

Discarding at consolidation is the cleanest form: information that does not meet the relevance threshold never enters long-term memory, by design. The relevance scorer is the gatekeeper, and its threshold is a deliberate policy choice.

Decay through disuse happens naturally if retrieval weights and recency boosts are designed correctly: unretrieved memories sink in the ranking until they are functionally invisible, without being deleted. This is an engineering analog of biological decay, and the analogy is imperfect — the memory remains intact and reachable on direct query; it simply does not surface in default ranking. Whether this constitutes "forgetting" in any meaningful sense is itself a question worth examining, and one a future paper may take up directly.

Explicit deletion is required for compliance, privacy, and correction of error. The architecture must support hard deletion of a memory and all of its derivatives — markdown file, embeddings, metadata, and ideally any cached retrievals — keyed by memory ID, by source ID, or by tenant. This is a place where the human-brain analogy breaks down: humans cannot reliably forget on demand, but AI systems must be able to, and the architecture must make it tractable.

---

5. Why This Shape Beats Flat RAG

It is worth being explicit about what this architecture buys, relative to the now-standard pattern of "LLM plus vector database."

A flat RAG system has no notion of session boundaries, so it cannot perform consolidation. It has no notion of supersession, so contradictions accumulate silently. It has no provenance chain, so it cannot defend its claims. It has no relevance scoring, so it remembers small talk with the same fidelity as decisions. It has no use-dependent reinforcement, so the most-needed memories are no easier to retrieve than the never-needed ones. It has no notion of forgetting that does not amount to deletion. And it conflates the canonical content of a memory with the index used to retrieve it, making the system brittle to embedding-model changes.

A layered cognitive architecture solves each of these by treating memory as a *process* rather than a *table*. Each layer has a clear job. Each transition between layers is an explicit cognitive operation. The architecture pays a real cost in complexity — five layers and an asynchronous worker is more than a vector database and a retriever — but it pays that cost in exchange for an agent that can demonstrably learn, update, supersede, and explain itself over time.

---

6. Open Questions and Honest Limits

This framework is not a solved problem. Several questions remain open and deserve research:

How should consolidation handle multi-session synthesis? A single insight may emerge from patterns across many sessions, none of which contain the insight in isolation. Per-session consolidation will miss this. A nightly deep-pass consolidator that re-examines the recent corpus is the obvious answer, but its design — what window, what synthesis strategy, what cost ceiling — is genuinely difficult.

How should the system avoid self-reinforcing error? Use-dependent reinforcement strengthens memories that are retrieved often. If an early-stored incorrect memory is retrieved often, its weight grows. Mechanisms for *contradicting evidence to weaken a memory*, not merely decay through disuse, are an active research area.

How do procedural memories actually work in this framework? Skills, workflows, and learned task patterns are different in kind from facts and events. Storing them as markdown documents is workable but feels like a loose fit. The proper home for procedural memory in an agent system is, in our judgment, an unsolved design question.

What is the right relevance threshold? Set it too high and the agent forgets things it later wishes it remembered. Set it too low and the long-term store fills with noise that degrades retrieval quality. The threshold is empirically determined per deployment, but principles for tuning it remain underdeveloped.

At what scale does ranking-as-deprioritization stop being sufficient? This framework treats forgetting as soft ranking decay rather than deletion, which is appropriate at the scale of an individual user's accumulated memories but may break down at organizational or larger scales. The point at which retrieval quality requires more aggressive measures — tiered storage, abstraction-driven compression, or selective deletion — is not well characterized.

These are real limits, and any team building on this framework should expect to encounter them. The argument of this paper is not that the architecture is finished; it is that *this is the right shape* for AI memory, and incremental progress within this shape is more productive than further refinement of flat RAG.

---

7. Conclusion

The human brain is the only known system that does memory well at the timescales and complexities relevant to agentic AI. It does so not with a single mechanism but with a coordinated set of stages — sensory buffering, working memory, consolidation, long-term storage, and active retrieval — each serving a distinct function and each operating on different timescales. AI agents that aspire to genuine continuity across interactions need an architecture that respects this decomposition, not one that collapses it into a single retrieval call.

The five-layer architecture presented here — sensory ingestion, short-term working memory, the consolidation bridge, long-term memory in three co-located stores, and active retrieval — is a direct translation of the brain's functional decomposition into engineering primitives that exist today. None of the components are exotic. The novelty is in their arrangement and in the discipline of treating memory as a process with stages, each of which deserves its own design.

Memory is what turns a model into an agent that can be trusted over time. Building it properly is worth the investment.

Changes from Version 1.0

This revision is light. The framework, structure, and arguments of v1.0 are unchanged. What v1.1 fixes:

- **Citations are now properly attributed.** The March 2026 survey is correctly attributed to Pengfei Du, the canonical Sumers et al. work is given its due, and the ACT-R reference points to Anderson (1983) where it should. v1.0 had several citations that were paraphrased or unattributed in ways that would not pass a careful read.
- **Quotes are sharpened.** Where v1.0 paraphrased the Du survey closely enough that the language could be confused for a direct quote, v1.1 either quotes precisely with attribution or paraphrases more clearly.
- **Section 3.5 clarifies the decay framing.** v1.0 used biological language ("the path becomes harder to find") for what is, operationally, a ranking-based deprioritization. v1.1 makes this distinction explicit: decay in this architecture means deprioritization in ranking, never deletion. The memory remains reachable on direct query.
- **Section 4.3 acknowledges that the engineering analog of biological decay is imperfect.** Whether ranking-based deprioritization constitutes "forgetting" in any meaningful sense is flagged as an open question rather than left implicit.
- **Section 6 adds one open question** about scale: at what point does ranking-based deprioritization stop being sufficient? This question was implicit in v1.0 and is now explicit.
- **Author attribution.** v1.0 had no byline. v1.1 makes the authorship and the AI research/drafting assistance transparent.

What v1.1 does *not* do: it does not retrofit later thinking about agent-portability, the substrate-vs-brain distinction, or the brain-core promotion model. Those evolutions belong in their own dedicated whitepapers, and v1.1 is kept self-contained so it can stand as the foundational reference for the body of work that follows.

References

Anderson, J. R. (1983). *A spreading activation theory of memory.* Journal of Verbal Learning and Verbal Behavior, 22(3), 261–295.

Atkinson, R. C., & Shiffrin, R. M. (1968). *Human memory: A proposed system and its control processes.* Psychology of Learning and Motivation, 2, 89–195. The foundational work on the multi-store model.

Baddeley, A. (2000). *The episodic buffer: A new component of working memory?* Trends in Cognitive Sciences, 4(11), 417–423.

Du, P. (2026). *Memory for Autonomous LLM Agents: Mechanisms, Evaluation, and Emerging Frontiers.* arXiv:2603.07670. Survey of memory mechanisms in LLM-based agents covering work from 2022 through early 2026; the source of the diagnostic quotes in §1.

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.* The source of the recency-relevance-importance retrieval scoring approach referenced in §3.5.

Squire, L. R. (2004). *Memory systems of the brain: A brief history and current perspective.* Neurobiology of Learning and Memory, 82(3), 171–177.

Sumers, T. R., Yao, S., Narasimhan, K., & Griffiths, T. L. (2024). *Cognitive Architectures for Language Agents.* Transactions on Machine Learning Research. The canonical modern reference for cognitive-architecture framings of LLM agents.

Tulving, E. (1972). *Episodic and semantic memory.* In E. Tulving & W. Donaldson (Eds.), Organization of Memory. Academic Press.

This whitepaper presents a framework for discussion and engineering reference. It is not tied to any specific implementation or product. The architecture described can be realized with a wide range of off-the-shelf components, and the choice of those components is appropriately deferred to deployment context.