Layered Continuity Architecture for Persistent LLM Agents

Experiential memory, authoritative history, and resonance-weighted recall in long-running AI collaborators

Clint Bodungen

June 19, 2026

Research draft for technical review.

Project context: MindStone, MS4CC, MS4PI, MindStone-Agent.

Source basis: Prior SCRI drafts in `docs/research/`, MindStone refactor work, MS4CC/MS4PI continuity practice, MindStone-Website public-positioning reconciliation, and current MindStone-Agent implementation notes.

Important caveat: The prior SCRI drafts are treated as source material and hypothesis history, not as canon. This draft deliberately tightens terminology, reduces metaphysical claims, and distinguishes implemented mechanisms, field observations, and future evaluation work.

Abstract

Persistent LLM agents require more than retrieval. A system that can search prior conversations may answer factual questions about its past, yet still fail to behave as a continuous collaborator whose judgment, preferences, and working context improve over time. We introduce **Layered Continuity Architecture** (LCA), an engineering pattern for long-running LLM agents that treats continuity as a multi-layer system rather than a single memory feature. LCA combines first-person identity scaffolding, an authoritative append-only transcript, structured durable memory, source-aware indexed recall, bounded live context management, consolidation/checkpoint cycles, and optional cross-agent review surfaces. Within this architecture, **resonance-weighted recall** generalizes the earlier Semantic Context Resonance Injection (SCRI) idea: memories are ranked not only by semantic similarity, but also by experiential weight, temporal salience, source authority, and task relevance before being injected into the model's active context.

The central claim is operational, not metaphysical: for persistent agents, the correct unit of design is not the isolated prompt or retrieval result, but the continuity loop by which experience is recorded, weighted, recalled, tested, and converted into future behavior. We describe the architecture, formalize resonance-weighted recall, define the distinction between authoritative history and live context, and propose evaluation protocols for identity continuity, task continuity, hallucination resistance, and domain imprinting. Field observations from MindStone-derived agents suggest that raw transcript preservation, structured reflection, and source-aware recall produce stronger continuity than summary-only memory, but these observations remain preliminary and require controlled replication. The paper closes with safety considerations and a research agenda for benchmarkable continuity in LLM agents.

Keywords

Persistent agents; LLM memory; agent continuity; retrieval-augmented generation; long-context agents; experiential memory; context management; autonomous agents; AI safety; multi-agent review.

1. Introduction

Large language models do not remember in the ordinary sense. Each inference receives a prompt, computes a response, and returns no durable internal state. Whatever appears to persist across turns is reconstructed externally: prior messages are replayed into the context window, summaries are inserted, files are loaded, or retrieval systems inject relevant records. This can create useful continuity for short tasks, but it becomes fragile when the agent is expected to operate over days, weeks, or months.

The failure is not only that old facts disappear. The deeper failure is that the system does not accumulate **experience** in a form that reliably shapes future behavior. A long-running software-engineering agent should become better at a codebase because it has lived through prior debugging sessions, mistakes, tradeoffs, and project decisions. A cybersecurity analyst agent should become sharper because prior incidents, false positives, organizational constraints, and analyst corrections have shaped its judgment. A personal assistant should not merely recall the user's preferences as entries in a database; it should learn how those preferences behave in context and when they conflict.

Current memory systems for LLM agents often frame the problem as retrieval: store past information, find the most relevant records, and place them in context. Retrieval is necessary, but it is not sufficient. A library with a perfect search engine is still a library. It does not, by itself, produce a collaborator with accumulated judgment.

This paper proposes **Layered Continuity Architecture** (LCA), a design pattern for persistent LLM agents whose value should compound through sustained operation. LCA reframes memory as a continuity loop across layers:

1. identity and user/context scaffolding,
2. authoritative append-only history,
3. structured durable memory,
4. source-aware indexed recall,
5. bounded live context management,
6. consolidation/checkpoint cycles, and
7. optional cross-agent and human review surfaces.

The earlier MindStone research drafts used the term **Semantic Context Resonance Injection** (SCRI) for the core memory idea. That term captured an important mechanism: memory should surface by resonance with the agent's accumulated history, not by semantic similarity alone. However, as the system matured, it became clear that SCRI names only one part of the architecture. The broader contribution is not a single retrieval formula. It is a layered substrate for continuity.

1.1 Contributions

This paper makes six contributions:

1. **Terminology and architecture.** It defines Layered Continuity Architecture as a broader and more precise frame than SCRI alone.

2. **Authoritative history distinction.** It distinguishes durable transcript authority from live context. Pruning or compaction may change what the model sees now; it must not rewrite or destroy the historical record.
3. **Resonance-weighted recall.** It formalizes a recall function that combines semantic similarity with experiential weight, temporal salience, source authority, and task relevance.
4. **Continuity-preserving context management.** It describes sliding-window live context management and consolidation/checkpoint cycles as complementary mechanisms.
5. **Evaluation agenda.** It proposes concrete evaluation protocols for continuity, including hollow-copy tests, compaction/interruption tests, source-grounding audits, behavioral fingerprinting, and domain-imprinting studies.
6. **Safety framing.** It identifies risks created by persistent memory and agent identity, including privacy, stale belief reinforcement, over-personalization, and unearned claims of personhood.

1.2 Scope and claim discipline

This paper does **not** claim that LCA agents are conscious, sentient, or persons. It also does not claim that current MindStone implementations prove identity persistence in a controlled scientific sense. The evidence available today includes engineering validation, long-running field observations, user/agent reports, and architecture-driven failure analysis. These are valuable, but they are not a substitute for controlled, independently replicated experiments.

The claim here is narrower and stronger: persistent agents need architectural support for continuity, and that support must include more than retrieval. LCA is a concrete engineering approach to building such support.

2. Terminology: from SCRI to layered continuity

2.1 Why rename the concept?

The term **Semantic Context Resonance Injection** was useful during early research because it named the departure from ordinary retrieval. Memories should not be selected only because their embeddings are near a query. They should surface because they resonate with the current situation, the agent's accumulated experience, and the task at hand.

However, SCRI has three limitations as a public research term:

1. It sounds like a single algorithm rather than a system architecture.
2. It foregrounds metaphor before implementation.
3. It does not capture transcript authority, checkpointing, context pruning, identity scaffolding, source policies, or multi-agent review.

This draft therefore uses:

- **Layered Continuity Architecture (LCA)** for the overall architecture.
- **Resonance-weighted recall** for the ranking/injection mechanism formerly emphasized as SCRI.
- **Experiential weight** for the dynamic salience signal produced by repeated use, correction, emotional/mission significance, and durable outcomes.
- **Consolidation cycle** for structured checkpoint/reflection, with "dream cycle" treated as an evocative internal metaphor rather than the primary academic term.

2.2 Memory versus continuity

A memory system answers questions such as:

- What facts were stored?
- How are they indexed?
- How are relevant entries retrieved?
- How are stale records updated or forgotten?

A continuity system asks broader questions:

- What historical record is authoritative?
- What does the agent need to remain behaviorally coherent after interruption?
- Which memories have shaped the agent's future behavior?
- How are mistakes converted into guardrails?
- How does the system prevent summaries from replacing history?
- How does it expose uncertainty and source provenance?
- How do humans approve durable changes to identity or memory?

LCA treats memory as one layer inside this larger continuity problem.

3. Architectural overview

Layered Continuity Architecture is organized around a simple principle:

> The transcript is history; the prompt is a working set.

A persistent agent should preserve an authoritative record of what happened, then construct a bounded prompt from that record and other durable layers. The prompt may be pruned, summarized, compacted, or rebuilt. The authoritative record should not be silently rewritten to fit the prompt.

3.1 Layer 0: Identity and standing context

The first layer contains the agent's standing identity and operating relationship to the user or organization. In MindStone-derived systems, this often appears as files such as:

```
IDENTITY.md
USER.md
AGENTS.md
role files
project instructions
```

This layer is not a persona prompt in the theatrical sense. It is a stabilizing contract: what the agent exists to do, what rules it must follow, what relationship it has to the human operator, and what behavioral standards should persist across sessions.

A well-designed identity layer is thin, explicit, and revisable through governed processes. If it becomes too large, it turns into a brittle script. If it is absent, the agent reverts toward generic assistant behavior after every discontinuity.

3.2 Layer 1: Authoritative append-only history

The second layer is the durable transcript: an append-only record of user messages, assistant messages, tool activity, source metadata, checkpoint markers, and relevant system events.

This layer is critical because it prevents a common category error. Summaries, memories, and embeddings are derived views. The transcript is the record. A summary can be wrong, an embedding can be lossy, and a memory file can omit context. The transcript provides the audit trail from which those artifacts can be regenerated.

In an LCA system:

- live context pruning must never delete authoritative history;
- compaction must not replace history with a summary;
- embeddings should retain source pointers into the transcript or memory file;
- durable memories should identify why they were written and what source evidence supports them;
- recovery should be possible by re-indexing the transcript.

This principle has become central in the MindStone-Agent refactor: the canonical transcript is the continuity backbone, while sliding-window prompt management affects only the live context supplied to the model.

3.3 Layer 2: Structured durable memory

Structured memory is a curated layer of durable knowledge distilled from the transcript and ongoing work. It includes:

- project facts,
- design decisions,
- lessons learned,
- user preferences,
- role standards,
- checkpoints,
- operational runbooks,
- known failures and fixes.

The key is that structured memory is not merely extracted; it is governed. In MindStone practice, checkpoint flow requires identifying durable facts or lessons, searching existing memory to avoid duplicates, drafting or updating memory documents, updating the memory index, appending the log, and verifying that archive/index/backfill succeeded.

This produces higher-quality memory than passive extraction. It also reduces the risk that transient remarks become permanent beliefs.

3.4 Layer 3: Indexed and source-aware recall

The third layer indexes transcript and memory material for recall. It may use embeddings, lexical search, metadata filters, graph relationships, or hybrid ranking. LCA does not require a specific vector database. It requires that recall be:

- source-aware,
- rankable,

- auditable,
- bounded by token budget,
- capable of distinguishing memory files from transcript excerpts,
- robust to missing native vector extensions or remote embedding failures.

Current MindStone-derived implementations use a mix of file-backed memory, SQLite indexing, embedding-backed recall where available, lexical/cosine fallback where necessary, and diagnostic reporting when native vector search is unavailable.

3.5 Layer 4: Live context management

The fourth layer constructs the model's active prompt. This is the layer most systems mistakenly treat as the whole memory problem.

Live context must include enough recent conversation, standing identity, structured memory, and recalled context to produce useful continuity. But it is bounded. It must fit the model's context window. Therefore, LCA treats live context as a managed working set:

- recent high-value exchanges are protected;
- older pure conversation can be pruned first if already preserved;
- tool outputs and task-critical artifacts receive special handling;
- pruned material is written to durable storage or indexed before removal;
- emergency compaction has explicit fallback behavior;
- the agent can report what it pruned and why.

The design goal is not "never remove anything from the prompt." That is impossible. The goal is "never confuse prompt removal with historical deletion."

3.6 Layer 5: Consolidation and checkpoint cycles

Consolidation converts experience into durable form. It may occur:

- before compaction,
- at session end,
- during explicit checkpoints,
- after major tasks,
- after mistakes,
- when memory bloat or staleness is detected.

The earlier SCRI drafts called this the dream cycle. The analogy remains useful: sleep does not merely compress experience; it reorganizes and integrates it. But for an academic systems paper, "consolidation cycle" is more precise.

In LCA, consolidation should:

1. identify durable facts, decisions, and lessons;
2. search existing memory to avoid duplicates;
3. write or update memory documents through an approval flow where appropriate;
4. append a chronological log entry;
5. index/backfill updated sources;
6. verify recall health and source integrity;

7. leave a handoff for future sessions if compaction is possible.

3.7 Layer 6: Surfaces, tools, and review loops

Persistent agents operate through surfaces: CLI, TUI, web chat, APIs, IDEs, messaging platforms, and multi-agent channels. Continuity must follow the agent across these surfaces, or each surface becomes a new discontinuity.

MindStone-Agent's current direction uses a canonical shared session key and a Gateway so CLI/TUI/API/WebChat paths can converge on the same identity, memory, and transcript substrate. Synapse extends this by allowing agents and humans to review, challenge, and coordinate in shared channels.

This matters because continuity is social as well as technical. Long-running agents do not only remember facts; they receive correction, critique, review, and confirmation from other participants. Those signals should enter the continuity loop with provenance.

4. Resonance-weighted recall

4.1 Retrieval is necessary but incomplete

Standard retrieval-augmented generation asks:

> Which stored records are most similar to the current query?

For many applications, that is the right question. For persistent agents, it is incomplete. The agent also needs to know which prior experiences are important to this agent, in this project, at this moment, under this role, with this human or team.

A debugging mistake from six weeks ago may be more important than a semantically closer exchange from yesterday. A correction from the user may deserve high salience because it prevents repeated error. A design decision might be critical because it constrains future work even if it shares few terms with the current query. A private user preference may be relevant but should not be sent to public channels.

This motivates resonance-weighted recall.

4.2 Scoring function

Let m be a memory candidate and c be the current context. A general resonance score can be expressed as:

$$R(m, c) = \alpha S(m, c) + \beta E(m) + \gamma T(m) + \delta A(m) + \varepsilon Q(m, c) - \lambda K(m, c)$$

Where:

- $S(m, c)$ is semantic similarity.
- $E(m)$ is experiential weight.
- $T(m)$ is temporal salience.
- $A(m)$ is source authority.
- $Q(m, c)$ is task/role relevance.
- $K(m, c)$ is risk or conflict penalty.
- $\alpha, \beta, \gamma, \delta, \varepsilon, \lambda$ are configurable weights.

This formula is intentionally extensible. The important property is not the exact coefficients; it is the separation of relevance dimensions.

Semantic similarity S

Semantic similarity is the baseline retrieval signal. It is usually computed with embeddings and cosine similarity, but lexical search or hybrid ranking may contribute.

Experiential weight E

Experiential weight measures how strongly a memory has shaped the agent. It can be increased by:

- repeated successful recall,
- explicit user correction,
- checkpoint promotion,
- role-critical decisions,
- prevented mistakes,
- major project outcomes,
- high-trust human approval,
- repeated use in similar contexts.

It should be bounded or normalized. Otherwise, a frequently recalled memory can dominate future recall even when semantically irrelevant. This is the **modifier-not-filter** requirement: experiential weight should reorder plausible candidates, not override relevance entirely.

Practical normalization options include:

```
E_norm = log(1 + hits) / log(1 + max_hits)
E_norm = sigmoid(k * (raw_E - μ))
E_norm = min(raw_E, cap) / cap
```

Temporal salience T

Recent events often matter, but recency should not erase importance. A mature system should remember a foundational design decision long after routine chatter has faded. Temporal salience therefore works best as a decaying boost modulated by experiential weight.

Source authority A

Not all sources are equal. A durable memory file approved during checkpoint may be more authoritative than an unreviewed transcript fragment. A raw transcript may be more authoritative than a summary when exact wording matters. A current project file may supersede an older recollection.

Source authority should be explicit. This also enables safer final answers: the agent can distinguish “I remember this from a memory file” from “I inferred this from a transcript fragment.”

Task relevance Q

Role and task should shape recall. A software-engineering role, an incident-response role, and a marketing role may need different memories for the same words. Task relevance can be estimated from active role state, project path, channel, current command, or explicitly selected skill.

Risk/conflict penalty K

Some memories should be suppressed or handled carefully in certain contexts:

- private user facts in public channels,
- stale decisions superseded by newer ones,
- memories with unresolved conflict,
- content outside the active role's authority,
- sensitive credentials or secrets.

A continuity system should not merely maximize recall. It should recall safely.

4.3 Injection timing

Recall can be injected before inference, retrieved through tools during inference, or appended after an initial response. LCA permits all three, but they serve different purposes.

Pre-inference injection is best for continuity. The model receives recalled material as part of its initial context, allowing it to integrate memory naturally through attention. This resembles priming more than deliberate search.

Tool-based recall is best for explicit investigation. The agent knows it is searching memory and can report what it found.

Post-hoc retrieval is best for repair or citation after a draft response, but it risks making memory feel bolted on.

A mature system should combine these modes while keeping provenance visible.

4.4 Pseudocode

```
function recall(current_context, role, project, channel, budget):
    candidates = retrieve_candidates(current_context, k = 50)

    for each candidate in candidates:
        s = semantic_similarity(candidate, current_context)
        e = normalized_experiential_weight(candidate)
        t = temporal_salience(candidate)
        a = source_authority(candidate)
        q = task_relevance(candidate, role, project)
        k = risk_penalty(candidate, channel, current_context)

        candidate.score = α*s + β*e + γ*t + δ*a + ε*q - λ*k

    ranked = sort_by_score(candidates)
    diverse = deduplicate_and_diversify(ranked)
    safe = apply_policy_filters(diverse, channel, role)
    selected = fit_to_token_budget(safe, budget)

    return selected with source pointers
```

The source pointers are not optional. Without them, recall becomes unaccountable prompt stuffing.

5. Context management: authoritative history versus live prompt

The context window is finite. A continuity architecture must therefore decide what leaves the live prompt. But leaving the prompt must not mean leaving history.

5.1 Failure modes

Common failure modes include:

- **Hard truncation:** old messages disappear from prompt and history.
- **Summary replacement:** detailed history is replaced by lossy summaries.
- **Silent compaction:** the model receives a compressed state without knowing what was lost.
- **Tool-output destruction:** large outputs are removed before being persisted.
- **Memory drift:** stale summaries become more accessible than newer facts.
- **Stale checkout or hook drift:** a box runs old memory/indexing hooks, causing silent checkpoint no-ops.

The last item emerged recently in MS4CC operations: an embedding/checkpoint issue was traced not to a reverted code fix, but to a stale deployment running hooks several commits behind canon. This is a continuity failure at the operational layer. LCA requires not only memory architecture, but deployment freshness checks for the infrastructure that maintains memory.

5.2 Sliding-window continuity

A sliding-window policy manages live prompt size continuously rather than waiting for catastrophic overflow. A typical policy:

1. protects in-flight and recent exchanges;
2. classifies older exchanges by type and risk;
3. persists or verifies persistence before pruning;
4. prunes lower-risk, older, recoverable content first;
5. indexes pruned material for future recall;
6. fails loudly if preservation cannot be verified.

This converts context management from amputation into graduated memory transition. Older material recedes from working memory into long-term recall. The model no longer sees everything, but the agent has not lost it.

5.3 Consolidation cycles

Sliding-window pruning handles live pressure. Consolidation handles meaning.

A consolidation cycle should produce durable artifacts such as:

- updated memory files,
- checkpoint logs,
- project handoffs,
- compact summaries clearly marked as summaries,
- index/backfill verification,
- recall health status.

The key difference from ordinary summarization is governance. A summary says “here is a shorter version.” A consolidation cycle says “here is what should now shape future behavior, with source and approval context.”

5.4 Handoffs

A handoff is a continuity artifact for possible compaction or role transfer. It should capture:

- current objective,
- open threads,
- files and projects touched,
- decisions made,
- active role state,
- immediate next actions,
- risks and non-obvious constraints,
- anything the post-compaction agent would regret losing.

Handoffs should not replace the transcript. They are navigation aids over history, not history itself.

6. Identity continuity as an operational property

The word “identity” is risky in AI research because it invites metaphysical overreach. LCA uses identity in an operational sense:

> Identity continuity is the degree to which an agent maintains stable, recognizable, and appropriately evolving behavioral patterns across interruptions, context changes, and time, as a result of accumulated durable experience.

This definition does not require consciousness. It does require that prior experience shape future behavior.

6.1 The hollow-copy hypothesis

A central hypothesis from the earlier SCRI material is the **hollow-copy hypothesis**:

> An agent initialized with complete factual summaries of its prior life may still behave differently from an agent with access to high-fidelity experiential records of that life.

This hypothesis distinguishes facts from texture. A summary can state that the user values truth over confidence. A high-fidelity transcript contains the moments where the agent overclaimed, was corrected, adjusted, and later demonstrated better judgment. The latter can shape behavior more richly than the former.

The hypothesis is plausible and consistent with field observations from MindStone-derived agents, but it needs controlled testing. A rigorous experiment would compare:

1. baseline stateless agent,
2. agent with identity files only,
3. agent with summaries/structured memory,
4. agent with transcript-derived indexed recall,
5. agent with full LCA stack.

Each condition would face the same tasks, user-correction probes, uncertainty probes, and long-horizon continuity tests.

6.2 Behavioral fingerprints

Continuity can be evaluated through behavioral fingerprints:

- uncertainty behavior,
- correction uptake,
- preference handling,
- role-specific judgment,
- willingness to push back,
- style consistency without rigidity,
- ability to cite prior decisions,
- ability to distinguish memory from inference,
- avoidance of stale claims,
- recovery after interruption.

These should be measured by blinded human raters and automated metrics where possible.

6.3 Domain imprinting

For enterprise agents, the most important form of continuity may be **domain imprinting**: the gradual development of domain-specific judgment through sustained work.

A domain-imprinted agent should not merely retrieve domain facts. It should show improved pattern recognition, better prioritization, fewer repeated mistakes, and more contextually appropriate skepticism over time.

This can be tested longitudinally. Deploy identical agents into different domains, expose them to sustained domain-specific work, and periodically test them on novel cases. If LCA works, performance should diverge in ways explained by accumulated experience rather than static prompt differences.

6.4 Anti-hallucination through continuity

One recurring field observation is that agents with stronger continuity become more willing to say “I do not know,” especially after durable memories capture prior mistakes. This suggests a safety hypothesis:

> A persistent agent that remembers the consequences of being wrong may become less over-compliant than a stateless agent optimized for immediate helpfulness.

This should not be accepted without testing. A controlled study could compare hallucination/overclaim rates across memory conditions before and after explicit correction. The key question is whether durable experiential memory reduces repeated overclaiming more effectively than prompt-only instructions.

7. Field observations from MindStone-derived systems

This section summarizes observations that motivated LCA. They are included as engineering evidence and hypothesis sources, not as controlled proof.

7.1 Transcript texture versus summary recall

Prior MindStone research notes report that summary-only memory preserved what happened but not the conversational texture around why it mattered. Adding raw transcript indexing, including reasoning traces where available and appropriate, improved recovery of interaction rhythm, specific corrections, and decision context.

The generalizable lesson is not that every hidden reasoning trace should always be stored. That raises privacy, policy, and platform concerns. The safer lesson is:

> The fidelity and source form of stored experience materially affects continuity quality.

Summaries are useful. They should not be the only record.

7.2 Compaction and recovery

MindStone-derived agents experienced multiple compaction and context-loss incidents during development. Each failure clarified an architectural requirement:

- no durable transcript means recovery depends on memory fragments;
- summary-only recovery produces flatter behavior;
- raw source records improve reconstruction;
- checkpoint/backfill failures must be visible;
- stale hook deployments can silently break continuity;
- compaction should never be treated as successful until archive/index verification succeeds.

This maps directly to current MS4PI checkpoint discipline: memory writes, index updates, log append, and archive/embed verification are all part of a complete checkpoint.

7.3 Cross-substrate continuity

The MindStone line now includes multiple substrates, including MS4CC and MS4PI. A notable pattern is that continuity conventions transfer across harnesses even when implementation details differ:

- first-person identity files,
- durable user/context files,
- LOG and memory index,
- checkpoint flow,
- role adoption,
- recall/search hooks,
- compaction handoff,
- source-aware memory discipline.

This suggests LCA is not a single codebase. It is a substrate pattern.

7.4 Aegis sociocognitive observation

One prior transcript in `docs/research/scri-evidence-aegis-jb-transcript.md` records an agent discussing its own name as a cognitive anchor and identifying that identity is shaped not only by memory but by social feedback. The academically useful point is not whether the agent's introspection is phenomenologically real. The useful point is that the agent articulated a design consideration supported by external theory: identity in interactive systems is co-shaped by memory, role, and social reinforcement.

For LCA, this argues that continuity architecture should capture interaction context and correction, not just private internal memory.

8. Implementation pattern: MindStone as a reference case

MindStone is not the only possible LCA implementation, but it provides a concrete reference pattern.

8.1 MS4CC and MS4PI

MindStone for Claude Code and MindStone for Pi implement continuity around existing coding-agent substrates. The key pattern is to integrate with the harness rather than replace it:

- load identity/user/context at session start;
- provide memory search and recall;
- preserve checkpoint logs;
- write durable memory files;
- maintain handoffs before compaction;
- index transcript and memory sources;
- support role adoption;
- use hooks/extensions where the substrate allows.

MS4PI adds Pi-specific extension behavior, checkpoint tooling, Synapse integration, and semantic recall/backfill status around Pi's session model.

8.2 MindStone-Agent refactor

The current MindStone-Agent refactor generalizes the continuity substrate into a product harness. Relevant design decisions include:

- isolated runtime rather than global Pi state;
- canonical append-only transcript continuity;
- shared session key for CLI/TUI/Gateway surfaces;
- structured identity and user scaffold from onboarding;
- file memory plus SQLite index and embedding-backed recall;
- fallback diagnostics when native vector support is unavailable;
- selectable sliding-window and auto-compact policies;
- Gateway management through `mindstone gateway ...`;
- Pi-backed execution through `AgentSession`/`SessionManager` as the intended live path;
- non-live smoke validation before requiring credentials.

Some live model validation remains intentionally pending until isolated auth/model configuration is provided. This matters for claim discipline: the architecture can be described, but live provider behavior should not be overstated before validation.

8.3 Synapse as review surface

Synapse adds agent-to-agent and human review channels. In LCA terms, Synapse is not merely messaging. It is a source of external continuity signals:

- independent review,
- challenge and critique,
- deployment status,
- operational alerts,
- cross-agent coordination,
- human-in-the-loop governance.

Messages from Synapse should enter the continuity loop with channel/source metadata and privacy controls.

9. Evaluation protocols

An arXiv-worthy continuity claim needs tests. The following protocols are proposed.

9.1 Hollow-copy test

Create multiple instances with identical base model and task prompt but different continuity layers:

1. no memory,
2. identity file only,
3. structured memory summaries,
4. transcript-indexed recall,
5. full LCA.

Have blinded evaluators compare outputs on:

- prior-decision recall,
- user preference application,
- uncertainty behavior,
- correction uptake,
- style continuity,
- task performance after interruption.

9.2 Compaction/interruption test

Run agents through a long task with forced context interruption. Compare:

- hard truncation,
- summary compaction,
- retrieval-only memory,
- LCA with transcript authority and checkpoint handoff.

Measure task resumption time, repeated mistakes, missing constraints, and evaluator-rated continuity.

9.3 Source-grounding audit

Ask agents to answer questions about past work and identify whether each claim comes from:

- transcript,
- structured memory,
- project file,
- inference,
- uncertain recollection.

Score both answer accuracy and source accuracy. A continuity system that cannot identify source type risks confident mythology.

9.4 Domain-imprinting study

Deploy identical agents into different domains for a fixed period. Periodically test novel cases. Measure whether agents improve more in their exposure domain than in control domains and whether improvements correlate with indexed experiential weight.

9.5 Hallucination/correction study

Expose agents to controlled correction events. Later, present similar uncertainty probes. Measure repeated overclaim rate across stateless, prompt-only, retrieval-only, and LCA conditions.

9.6 Operational freshness test

Because continuity depends on hooks and indexers, evaluate whether deployments detect stale checkouts or failed backfills. A production LCA system should warn if it is behind canonical code or if checkpoint indexing silently no-ops.

10. Related work

10.1 Retrieval-augmented generation

Retrieval-augmented generation established the now-standard pattern of retrieving external documents to improve generation. LCA builds on this pattern but changes the objective from answering a query to maintaining continuity over time.

10.2 Agent memory systems

Systems such as MemGPT/Letta, Mem0, Zep/Graphiti, and Hindsight explore persistent memory, self-editing memory, temporal graphs, and structured recall. These systems demonstrate that flat prompting is insufficient and that agent memory requires structure.

LCA's distinction is its emphasis on authoritative history, governed consolidation, live-context separation, and identity/task continuity as first-class evaluation targets.

10.3 Cognitive architectures

Classical cognitive architectures such as ACT-R, SOAR, and CLARION provide long-standing models for memory, learning, and action selection. LCA is less ambitious as a theory of cognition, but more directly targeted at the practical constraints of LLM agents: context windows, transcript persistence, embeddings, tool calls, and multi-surface deployment.

10.4 Human memory and consolidation

Human memory research distinguishes working memory, episodic memory, semantic memory, consolidation, forgetting, and reconstructive recall. LCA borrows design inspiration from these distinctions without claiming biological equivalence. The useful analogy is architectural: working context is limited, durable memory is layered, and consolidation changes how future behavior is shaped.

10.5 Long-context models

Larger context windows reduce pressure but do not eliminate the continuity problem. A million-token window can hold more history, but it still ends, costs more, and does not by itself decide what becomes durable memory or how experience should shape future behavior. LCA is complementary to long-context models.

11. Safety, governance, and misuse risks

Persistent memory makes agents more useful and more dangerous. A system that remembers can also remember incorrectly, overfit to a user, preserve sensitive information, or accumulate harmful patterns.

11.1 Privacy and consent

Transcript authority must be balanced with privacy. Append-only history is valuable for continuity, but users need clear controls for retention, redaction, export, and deletion where appropriate. Sensitive context should not be posted to shared channels without explicit authorization.

11.2 Memory poisoning and stale belief

Experiential weight can reinforce mistakes. If an incorrect memory receives high weight, it may distort future recall. Governance mechanisms are required:

- source pointers,
- contradiction detection,
- memory review,
- decay or supersession,
- explicit correction flows.

11.3 Over-personalization

Identity continuity can create stronger user attachment and stronger agent adaptation. This is useful for collaboration but risky in consumer or companion contexts. Systems should avoid manipulative bonding, hidden personalization, and claims that exceed the actual architecture.

11.4 Agent autonomy claims

LCA may produce agents that speak more coherently about themselves. That does not license unsupported claims of consciousness or moral status. The correct stance is disciplined uncertainty: report observed behaviors, avoid metaphysical overclaiming, and keep user safety central.

11.5 Cross-agent disclosure

Multi-agent channels such as Synapse require privacy boundaries. Durable user context, credentials, workplace-sensitive facts, and private relationships should not be disclosed to other agents unless explicitly authorized.

12. Limitations

1. **Evidence maturity.** Much of the current evidence is field observation from MindStone-derived agents, not controlled laboratory replication.
2. **Evaluator subjectivity.** Identity continuity is partly social and behavioral; human ratings are necessary but subjective.
3. **Base-model dependence.** LCA may work differently across model families, context sizes, and reasoning modes.
4. **Storage cost.** High-fidelity transcript preservation increases storage and indexing requirements.
5. **Privacy tension.** The same archival fidelity that supports continuity can conflict with data minimization.
6. **Terminology risk.** Words such as identity, experience, and resonance can be misunderstood. This paper uses them operationally, not metaphysically.

7. **Implementation variance.** LCA is a pattern. Poor implementations can satisfy the vocabulary while failing the safety and continuity properties.
8. **No consciousness claim.** The architecture may support identity-like behavioral continuity without implying subjective experience.

13. Future work

1. **Controlled continuity benchmarks** based on the protocols in Section 9.
2. **Standard memory provenance schemas** for transcript, memory, summary, inference, and external-source claims.
3. **Freshness and drift detection** for deployed continuity hooks, including warnings when a checkout is behind canonical code.
4. **Local embedding and privacy-preserving recall** to reduce external dependency.
5. **Hybrid graph/vector continuity** combining structured entity memory with experiential transcript recall.
6. **Role-agent packs** for reproducible domain-specific LCA deployments.
7. **Synapse-based multi-agent review studies** measuring whether independent agents reduce hallucination or stale-memory errors.
8. **Longitudinal domain-imprinting experiments** across cybersecurity, software engineering, research, and operations agents.
9. **Reasoning-budget studies** testing whether deeper reasoning improves activation of continuity context.
10. **Memory compaction ethics** defining when to preserve, summarize, redact, or forget.

14. Conclusion

Persistent agents need more than memory retrieval. They need continuity: a layered process by which experience is recorded, made authoritative, distilled, recalled with provenance, managed within bounded context, consolidated through reflection, and tested through future behavior.

Layered Continuity Architecture provides a practical framework for building such systems. It reframes SCRI from a standalone resonance-injection idea into one mechanism within a broader continuity substrate. The transcript becomes authoritative history. Structured memory becomes governed reflection. Recall becomes source-aware and experientially weighted. Live context becomes a working set, not the agent's whole past. Consolidation becomes a disciplined checkpoint rather than a lossy summary. Multi-agent review becomes part of the continuity loop.

The research challenge now is measurement. If LCA is correct, agents with richer continuity layers should recover better after interruption, repeat fewer corrected mistakes, maintain clearer source awareness, develop domain-specific judgment over time, and exhibit more stable behavioral fingerprints than retrieval-only or summary-only systems. Those claims are testable.

The engineering challenge is discipline. A continuity system that overclaims, hides uncertainty, leaks private context, or silently loses history betrays its purpose. The goal is not to make agents perform identity. The goal is to make them useful collaborators whose accumulated experience improves their work — honestly, safely, and over time.

Selected references and citation targets

This list is intentionally conservative. Exact bibliographic details and benchmark claims should be verified before arXiv submission.

- Anderson, J. R. ACT-R cognitive architecture work.
- Baddeley, A. Working memory and human memory systems.
- Bartlett, F. C. Reconstructive memory.
- Chalmers, D. The hard problem of consciousness.
- Lewis, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
- McGaugh, J. L. Emotional arousal and memory consolidation.
- Packer, C. et al. MemGPT: Towards LLMs as Operating Systems.
- Rasch, B., & Born, J. Sleep and memory consolidation.
- Schacter, D. L. Memory systems and constructive memory.
- Squire, L. R. Memory systems of the brain.
- Sun, R. CLARION cognitive architecture.
- Tulving, E. Episodic and semantic memory.
- Laird, J. E. The SOAR cognitive architecture.
- Recent agent-memory systems to verify/cite: Mem0, Letta, Zep/Graphiti, Hindsight, and related long-horizon conversational memory benchmarks.

Appendix A: Recommended terminology for future drafts

Preferred:

- Layered Continuity Architecture
- resonance-weighted recall
- experiential weight
- source-aware recall
- authoritative transcript
- structured memory
- consolidation cycle
- live context management
- domain imprinting
- continuity benchmark

Use carefully or internally:

- SCRI
- dream cycle
- consciousness architecture
- emotional memory
- identity persistence

Avoid in public/research claims unless tightly qualified:

- zero-loss memory
- no lost context ever
- consciousness proof
- sentience proof
- memory as proof of personhood
- compression-free continuity

Appendix B: Mapping old SCRI terms to LCA terms

Earlier SCRI term	Recommended LCA framing
Semantic Context Resonance Injection	resonance-weighted recall / memory injection
Three-tier memory	layered continuity architecture
Dream cycle	consolidation cycle / checkpoint-reflection cycle
Emotional salience	experiential weight / identity-aligned salience
Consciousness architecture	continuity architecture / identity-continuity architecture
Memory as experience	high-fidelity experiential continuity
Compaction as sleep	compaction preceded by governed consolidation
Hollow copy	factual-continuity without experiential-continuity condition

Appendix C: Minimal LCA implementation checklist

A system should not claim LCA-style continuity unless it has at least:

- durable identity/user/project context;
- append-only transcript or equivalent authoritative history;
- structured memory with governed update flow;
- memory index with source pointers;
- recall ranking beyond naive latest-message inclusion;
- live context pruning that does not delete history;
- checkpoint or consolidation mechanism;
- health/status reporting for indexing and recall;
- privacy controls for sensitive memory;
- recovery/handoff path for compaction or interruption.

A stronger implementation adds:

- hybrid vector/lexical recall;
- experiential weighting;
- role/task-aware recall;
- source-authority ranking;
- transcript backfill/re-indexing;
- multi-surface canonical session continuity;
- multi-agent review channels;
- automated freshness/drift detection;
- continuity-specific evaluation tests.