

# Beyond Flat Memory: Sovereign Temporal Graphs and Emergent User Topology for AI Memory Governance

Richard “Ricky” Valentine  
Independent Researcher, Houston, TX, USA  
[Ricky@RichardValentine.dev](mailto:Ricky@RichardValentine.dev)

Version 1.0 · March 2026

## Abstract

We present the Sovereign Temporal Graph with Emergent User Topology (STG-EUT), a memory governance architecture that combines contamination-aware sovereign governance with emergent per-user knowledge topology. Current AI memory systems store user knowledge in flat, ungoverned structures where all items carry equal retrieval weight, contamination sources are architecturally indistinguishable, and organizational categories are predefined by system designers rather than derived from individual users. This paper identifies the Dual Contamination Problem—a self-reinforcing failure mode in which pre-existing data pollution and AI-generated errors coexist in the same store and amplify each other—and presents an architecture addressing it through four mechanisms:

1. *Sovereign Temporal Graph Governance*—a five-stage pipeline (ingest, detect, quarantine, validate, store) with tripartite provenance classification, a human-gated validation boundary structurally excluding ML from final governance decisions, and freeze/revert temporal state control.
2. *Emergent Per-User Topology*—unsupervised community detection over human-verified foundational memories producing thematic clusters unique to each user, with a dynamic lifecycle (formation, merging, weakening, collapse).
3. *Dual-Function Identity-Aware Retrieval*—emergent clusters simultaneously narrow retrieval scope and constitute the system’s evolving structural model of the user’s enduring concerns.

STG-EUT builds on Crystallization Theory (CT), extending the  $W(k, t)$  scoring function with sovereign governance semantics and emergent topology primitives. The contributions are architectural. A prototype case study validates structural plausibility—demonstrating governed ingestion, category-level topology divergence across user profiles, scoped retrieval improvement, and lifecycle dynamics under perturbation—while full graph-based community detection and multi-user longitudinal evaluation remain future work.

*Keywords:* knowledge governance, AI memory, contamination detection, emergent topology, per-user clustering, sovereign data, identity-aware retrieval, temporal graphs, MCP, knowledge lifecycle

*Related Patents:* #63/953,509 (Crystallization Theory), #63/962,609 (KEOS), #64/020,880 (STG-EUT)

---

## 1. Introduction

Large language model applications increasingly depend on persistent memory to maintain continuity across sessions, accumulate user context, and deliver personalized responses. The dominant architectural pattern for this memory is remarkably uniform: a vector store indexes conversation fragments as embeddings, a retrieval module surfaces the top- $k$  most similar fragments at query time, and the language model consumes them alongside the current prompt. Variations exist—some systems extract structured facts into key-value stores<sup>[1]</sup>, others layer knowledge graphs over raw embeddings<sup>[2, 3]</sup>, and a recent line of work introduces tiered storage analogous to an operating system’s memory hierarchy<sup>[4]</sup>—but all share a foundational assumption: every stored item occupies the same organizational plane, differentiated only by a similarity score computed at retrieval time.

We call this the *flat memory* assumption. Under flat memory, a user’s most identity-defining professional commitment and a passing mention of lunch carry equal structural weight. The system has no mechanism to distinguish foundational knowledge from ephemeral context except through the transient arithmetic of cosine similarity. There is no permanence, no emergence, and no organizational structure that reflects who the user is rather than what the user most recently said.

This paper argues that flat memory is not merely a performance limitation but a governance failure, and that the failure has two distinct dimensions that current architectures address—at best—in isolation.

The first dimension is what we term the *Dual Contamination Problem* (DCP). Enterprise knowledge ecosystems contain pre-existing data pollution: inconsistent formats across source systems, stale records that were never retired, contradictory entries created by different teams at different times. When an AI memory system ingests this data, it does not flag the inconsistencies—it embeds them with the same confidence as verified facts, propagating contamination downstream with the authority of a retrieval system that returned them as relevant. Simultaneously, the AI system itself introduces a second contamination vector: hallucinated facts, fabricated details, and confidently incorrect inferences that are written back into the memory store indistinguishably from legitimate entries. These two vectors are self-reinforcing. Pre-existing contamination increases the probability of AI-generated contamination (the model is more likely to hallucinate when its retrieved context is already inconsistent), and AI-generated contamination degrades the substrate from which future retrievals draw, amplifying pre-existing errors in subsequent cycles.

The critical observation is that DCP is not a model-level problem solvable by better guardrails or output filtering. It is a *system-architecture* problem: the memory store itself has no mechanism to distinguish contamination sources, no quarantine boundary between verified and unverified knowledge, and no governance authority that can freeze, revert, or audit the provenance of stored items. Hallucination detection addresses one vector. Data quality tooling addresses the other. Neither addresses the compound case where both contamination sources coexist in the same store and amplify each other.

The second dimension is the *flat topology* problem. Even if contamination were fully resolved, current memory systems would still fail to organize retained knowledge in a way that reflects the user’s identity. A physician, a patent attorney, and a music producer interacting with the same memory system develop identical structural representations: flat lists of extracted facts, perhaps tagged with timestamps or importance scores, but sharing the same universal schema. No system produces an organizational structure that is unique to each user—one where the physician’s memory self-organizes around clinical, research, and teaching themes while the producer’s memory self-organizes around composition, collaboration, and licensing themes. The categories themselves should be emergent properties of each user’s foundational knowledge, not developer-defined tem-

plates applied uniformly.

Existing work on provenance, temporal evolution, and state governance addresses the first dimension partially. Temporal knowledge graph systems [2, 3] track how facts evolve over time but provide no contamination handling or user-specific organization. Provenance graph systems [5, 6] maintain lineage metadata but do not distinguish AI-generated contamination from source-system contamination. Snapshot and versioning systems [7] offer rollback primitives but lack governance boundaries that would prevent unverified content from overwriting verified knowledge.

Separately, work on personalization and retrieval organization addresses the second dimension partially. Personalized memory systems [1, 4, 9, 21] retain user context but use predefined extraction schemas identical across all users—and none maintains a governance layer preventing AI-generated content from silently modifying human-verified knowledge. Corpus-level clustering systems [10, 11] demonstrate that emergent community structure can improve retrieval, but apply clustering per-corpus rather than per-user, producing identical structures for all users querying the same dataset.

No surveyed system combines contamination-aware sovereign governance with emergent per-user topology built from human-verified foundational memory. This paper presents such an architecture.

We introduce the *Sovereign Temporal Graph with Emergent User Topology* (STG-EUT), a memory architecture organized around two interlocking components. The first is a sovereign temporal graph that ingests knowledge through a five-stage pipeline—ingest, detect, quarantine, validate, store—with a hard constraint: no item may achieve verified status without explicit human confirmation. The graph maintains tripartite provenance (human-verified, source-system-inherited, AI-generated) with contamination boundary enforcement and freeze/revert temporal governance. The second is an emergent per-user topology layer: when a user designates verified memories as foundational ("Core memories"), the system constructs a per-user knowledge graph and applies unsupervised community detection to produce thematic clusters unique to that user. These clusters follow a dynamic lifecycle and serve a dual function: they narrow retrieval scope (a retrieval optimization) and constitute the system's evolving structural model of the user's enduring concerns (an identity perception layer).

The causal logic of the architecture is deliberate. The Dual Contamination Problem motivates the need for sovereign governance. Sovereign governance produces a substrate of trusted, human-verified knowledge. Trusted knowledge enables meaningful emergent topology—clustering unverified content would merely organize noise. And emergent topology enables identity-aware retrieval that is both faster and more personally relevant than flat vector search.

This paper makes four contributions:

1. **The Dual Contamination Problem.** We formalize a previously unnamed systems failure in AI-augmented knowledge environments: bidirectional, self-reinforcing contamination where pre-existing data pollution and AI-generated errors coexist in the same store and amplify each other. We distinguish this from model-level hallucination detection and from traditional data quality management, arguing that neither alone addresses the compound case.
2. **Sovereign Temporal Graph Governance.** We present a memory governance architecture with tripartite provenance classification, a quarantine stage isolating contaminated items from verified knowledge, a hard human-gated validation boundary excluding machine learning from final governance decisions, and freeze/revert temporal state control. The defining constraint—that no automated process may promote knowledge to verified status—is architectural rather than policy-based.
3. **Emergent Per-User Topology.** We present an architecture in which unsupervised community detection over human-verified foundational memories can produce thematic clusters

unique to each user, and we show prototype category-level divergence consistent with that claim. We define a dynamic lifecycle for these clusters (formation, merging, collapse) and are not aware of prior architectures in which the organizational structure of memory is itself an emergent, user-specific property rather than a developer-defined schema.

4. **Dual-Function Identity-Aware Retrieval.** We show that the proposed topology is designed to support both retrieval scoping and identity modeling as architecturally inseparable functions, and we provide prototype evidence consistent with this dual-function role.

The remainder of this paper is organized as follows. Section 2 surveys related work across temporal knowledge graphs, provenance systems, personalized AI memory, and corpus-level clustering. Section 3 formalizes the Dual Contamination Problem. Section 4 presents the Sovereign Temporal Graph governance architecture. Section 5 introduces emergent per-user topology. Section 6 describes identity-aware retrieval. Section 7 presents prototype results. Section 8 discusses limitations and future directions.

## 2. Related Work

This section surveys five classes of systems adjacent to STG-EUT. Each class addresses a subset of the problems this paper targets—temporal knowledge evolution, provenance tracking, state governance, personalized memory, or retrieval organization—but none combines contamination-aware sovereign governance with emergent per-user topology built from human-verified foundational memory. We organize the survey by system class rather than by individual paper, focusing on what each class solves and what it structurally cannot.

### 2.1 Temporal Knowledge Graphs

Temporal knowledge graphs maintain timestamped facts and track entity/relationship evolution. Zep’s Graphiti <sup>[2]</sup> builds per-user temporal graphs with bi-temporal modeling. Temporal RDF extensions <sup>[18]</sup> and event-centric knowledge graphs <sup>[19]</sup> provide formal representations of time-varying assertions.

What temporal KGs do not provide is *governance*. A temporal KG records that a fact changed at time  $t$ , but does not classify *why*—whether it reflects a genuine update, a correction, an AI-generated assertion overwriting a human-verified one, or contamination propagation. The temporal axis tracks evolution but does not distinguish contamination from correction, and does not produce per-user organizational topology.

### 2.2 Provenance Graph Systems

Provenance systems track data lineage: where items originated, how they were derived, what transformations they underwent. The W3C PROV standard <sup>[5]</sup> provides a formal ontology; blockchain-based provenance systems <sup>[6]</sup> offer tamper-evident records.

Provenance solves the attribution problem but not the governance problem. Recording that an item originated from an AI system is different from *preventing* that item from silently overwriting human-verified knowledge. Provenance is descriptive; governance is prescriptive. No surveyed provenance system distinguishes pre-existing data contamination from AI-generated contamination.

### 2.3 Snapshot and Versioned Graph Systems

Systems like Dolt <sup>[7]</sup> apply Git-style version control to data; RDF archiving approaches <sup>[20]</sup> maintain timestamped snapshots. STG-EUT’s freeze/revert (§ 4.5) shares surface-level mechanisms with

these systems. The distinction is that STG-EUT’s primitives operate within a governance context: a freeze is a governance lock, not a development workflow tool; a revert is a governance correction that preserves the contamination in the audit history.

## 2.4 Personalized AI Memory Systems

This class is the most directly relevant to STG-EUT and requires the most careful distinction. Multiple systems now offer persistent memory for AI assistants, each with different architectural choices.

**ChatGPT memory** <sup>[1]</sup> stores  $\sim 1,200$ – $1,750$  words of flat key-value pairs injected into the context window. All memories carry equal weight with no provenance tracking, contamination handling, or mechanism for designating memories as foundational.

**MemGPT/Letta** <sup>[4]</sup> introduces two-tier storage: "core memory" blocks always in context, and "archival memory" searchable on demand. Letta’s core memory is the closest prior concept to STG-EUT’s Core memories, but with a critical difference: Letta’s core blocks are developer-defined templates identical across all users, with no emergent topology or dynamic lifecycle. The agent manages its own memory, creating the self-validating loop that STG-EUT’s negative-ML constraint prevents.

**Mem0** <sup>[21]</sup> uses a hybrid vector/graph/key-value architecture with priority scoring, dynamic forgetting, and memory consolidation—the most architecturally advanced production system surveyed. However, its six extraction categories are predefined and identical for every user, its knowledge graph does not apply community detection for emergent clusters, and it has no human-gated governance boundary.

**MemoryBank** <sup>[9]</sup> introduces Ebbinghaus-inspired forgetting curves but applies decay uniformly with no permanence exceptions. **A-MEM** <sup>[8]</sup> (NeurIPS 2025) introduces Zettelkasten-inspired self-organizing memory—the strongest prior art for emergent structure—but operates per-agent without cross-user divergence, importance tiers, or governance. **Generative Agents** <sup>[13]</sup> established importance-weighted retrieval with decay but operates on a flat stream with no clustering or per-user divergence.

## 2.5 Corpus-Level Clustering and Retrieval Systems

**GraphRAG** <sup>[10]</sup> applies the Leiden community detection algorithm to a corpus-level knowledge graph, producing hierarchical community structures that guide retrieval through a DRIFT (Dynamic Reasoning with Iterative Filtering and Traversal) search process. GraphRAG demonstrates that emergent community detection can meaningfully improve retrieval over flat vector search—a finding that informs STG-EUT’s Island-guided routing (§ 6). The distinction is three-fold: GraphRAG’s communities are per-corpus (identical for all users), unfiltered (no governance or contamination handling on the indexed content), and single-function (retrieval only, with no identity-modeling capability). STG-EUT applies community detection per-user, over a governed substrate of human-verified foundational memories, to produce structures that serve both retrieval and identity perception.

**RAPTOR** <sup>[11]</sup> constructs hierarchical retrieval trees through recursive abstractive processing: document chunks are clustered, clusters are summarized, and the process repeats to build a multi-level index. RAPTOR demonstrates that cluster-first retrieval can outperform flat search. Like GraphRAG, RAPTOR operates per-corpus with a static hierarchy computed at index time. The hierarchy does not vary across users, does not evolve without full reindexing, and does not model user identity.

Table 1: Coverage of architectural properties across surveyed system classes.

| System Class                          | Temporal Evolution | Prov.      | Gov. Boundary | Per-User Topology | Id.-Aware Retrieval |
|---------------------------------------|--------------------|------------|---------------|-------------------|---------------------|
| Temporal KGs [2, 3, 18, 19]           | Yes                | Partial    | No            | No                | No                  |
| Provenance Systems [5, 6]             | Partial            | Yes        | No            | No                | No                  |
| Snapshot/Versioning [7, 20]           | Yes                | Partial    | No            | No                | No                  |
| Personalized Memory [1, 4, 9, 13, 21] | Partial            | No         | No            | No                | No                  |
| Corpus-Level Clustering [10, 11]      | No                 | No         | No            | No                | Partial             |
| <b>STG-EUT (this work)</b>            | <b>Yes</b>         | <b>Yes</b> | <b>Yes</b>    | <b>Yes</b>        | <b>Yes</b>          |

## 2.6 Gap Statement

The five system classes surveyed each contribute mechanisms relevant to the problems STG-EUT addresses: temporal evolution, provenance tracking, state governance, personalized memory, and retrieval organization. Table 1 summarizes the coverage.

No surveyed system combines contamination-aware governance (distinguishing pre-existing from AI-generated contamination with a human-gated validation boundary) with emergent per-user topology (user-specific thematic clusters formed from verified foundational memories with a dynamic lifecycle) and dual-function identity-aware retrieval (topology serving simultaneously as retrieval scope and identity model). Individual mechanisms exist across the landscape; the architectural combination does not.

## 3. The Dual Contamination Problem

The introduction presented the Dual Contamination Problem informally as two self-reinforcing contamination vectors. This section formalizes the problem, defines its components precisely, and explains why it is a systems-architecture concern that cannot be resolved at the model level alone.

### 3.1 Pre-Existing Contamination

The first vector originates in source systems. Enterprise data ecosystems contain format inconsistencies from schema migrations, stale entries never retired, contradictory values from independent update cycles, and orphaned references. This contamination is *pre-existing*: present before any AI system processes it. Traditional data quality tooling addresses some errors, but residual contamination persists in most production environments. When an AI memory system ingests this data, it inherits the contamination—a vector store embeds clean and stale records with equal fidelity, and a knowledge graph extraction pipeline does not flag contradictions unless specifically instrumented.

### 3.2 AI-Generated Contamination

The second vector originates in the AI system itself. Language models hallucinate during extraction, summarization, and inference. These errors are *AI-generated*: they did not exist in the source data and were introduced by the processing system. When written back into the memory store, they become indistinguishable from verified knowledge. Absent explicit provenance classification and governance boundaries, these items enter the same retrieval substrate as trusted content.

### 3.3 Self-Reinforcing Dynamics

The two contamination vectors interact. The interaction is what makes the Dual Contamination Problem a compound systems failure rather than two independent quality issues.

*Pre-existing contamination amplifies AI-generated contamination.* When a language model retrieves context from a memory store that contains stale or contradictory records, its outputs are more likely to be incorrect. The model may resolve contradictions by selecting the wrong alternative, propagate a stale fact as current, or generate inferences that are logically valid given the contaminated context but factually wrong. The contaminated context raises the base rate of AI-generated errors.

*AI-generated contamination degrades future retrieval quality.* When AI-generated errors are written back into the memory store, they become part of the context for subsequent retrievals. Future queries may surface these errors as relevant results, feeding them into the next processing cycle. The store’s contamination level increases over time as each cycle deposits new AI-generated errors alongside the pre-existing ones.

The result is a positive feedback loop: contamination in the store produces more contamination from the model, which is deposited back into the store, which produces more contamination in the next cycle. Each iteration degrades the store’s overall quality while simultaneously making the degradation harder to detect—because the AI system processes contaminated and clean content with equal confidence, and the store provides no structural signal distinguishing the two.

### 3.4 Why This Is a Systems-Architecture Problem

A natural response to DCP is to address each vector at its own level: apply data quality tooling to source systems (fixing pre-existing contamination) and apply guardrails to model outputs (catching AI-generated contamination). Both interventions are useful and neither is sufficient.

Data quality tooling operates on source systems before ingestion. It cannot address AI-generated contamination, which is introduced after ingestion. Output guardrails operate on model responses at generation time. They cannot address pre-existing contamination, which is already in the store. Neither intervention operates at the *memory store level*—the point where both contamination sources coexist and interact. The store itself remains architecturally unaware of the distinction between contamination sources.

The architectural gap is specific: existing memory systems do not maintain a *governance layer* at the storage level that (a) classifies stored items by contamination source, (b) enforces boundaries preventing cross-contamination between classified items, and (c) requires human verification before any item achieves trusted status. Without these architectural properties, the self-reinforcing loop described in § 3.3 operates unchecked. The STG governance architecture presented in § 4 is designed to fill this gap.

### 3.5 Illustrative Scenario

Consider a healthcare system ingesting patient data from an EHR and clinical notes from a language model assistant. The EHR contains a medication record correct when entered in 2021 but not updated since—the dosage was changed in a follow-up visit documented only in a paper chart. This is pre-existing contamination.

The language model processes a clinical note mentioning current medications and, drawing on the stale EHR record as context, generates a summary asserting the original dosage. This is AI-generated contamination—a confident inference produced from contaminated context. In a subsequent session, a clinician retrieves both the stale record and the AI-generated summary, which mutually reinforce the incorrect dosage. The pre-existing error has been laundered through AI-generated confirmation into apparent corroboration.

A governance architecture with tripartite provenance and contamination boundaries would in-

interrupt this loop at multiple points. The dynamic is structurally general: it operates wherever AI systems ingest from imperfect environments and write results back into the same store.

## 4. Sovereign Temporal Graph Governance

The Sovereign Temporal Graph (STG) is a governed memory substrate designed to resolve the Dual Contamination Problem through architectural constraints rather than heuristic filtering. Where existing memory systems treat ingestion as a write operation—content enters the store and becomes immediately retrievable—the STG treats ingestion as a governance event that must pass through a structured pipeline before any content achieves the status of verified knowledge. This section presents the five-stage governance pipeline, the provenance classification system, the structural constraints that enforce governance boundaries, and the temporal state primitives that enable rollback and audit.

### 4.1 Five-Stage Governance Pipeline

Every knowledge item entering the STG passes through five sequential stages. No stage may be bypassed, and no item may advance to a subsequent stage without satisfying the exit conditions of its current stage.

**Stage 1: Ingest.** The system receives a candidate knowledge item from one of several source classes: direct user input through a conversational interface, automated extraction from connected source systems (databases, document repositories, APIs), or AI-generated content produced by a language model during a prior interaction. At ingestion, the item is tagged with source metadata (originating system, extraction method, timestamp) and assigned a preliminary provenance classification based on its source class. Critically, ingestion does not confer any trust status. An ingested item is a candidate, not knowledge.

**Stage 2: Detect.** The contamination detection stage classifies each ingested item along two independent dimensions. First, it evaluates whether the item is consistent with existing verified knowledge in the graph—flagging contradictions, format mismatches, and temporal anomalies (e.g., a record claiming a state that conflicts with a more recently verified state). Second, it evaluates the item’s provenance classification for contamination risk: items originating from AI-generated content receive elevated scrutiny, items from source systems with known data quality issues receive moderate scrutiny, and items from human-verified sources receive minimal additional processing. The detection stage produces a contamination risk assessment but does not make a final determination. Detection is computational; governance is not. The role of detection is triage rather than adjudication: it identifies candidates for governance action but does not determine truth status.

**Stage 3: Quarantine.** Items flagged by the detection stage are placed in isolated storage partitions. Quarantined items are not retrievable by the main retrieval system and cannot influence query results. The quarantine stage enforces physical separation—quarantined items are stored in a distinct partition with no read path from the verified knowledge store. This prevents a failure mode common in existing systems where unverified or contradictory content "leaks" into retrieval results because it shares a vector space with verified content. Items that pass detection without flags proceed to Stage 4 without entering quarantine.

**Stage 4: Validate.** This is the governance boundary. Candidate items—whether arriving from quarantine or directly from detection—are presented to a human operator through a validation interface. The operator reviews the item’s content, its provenance metadata, its contamination risk assessment, and any conflicts with existing verified knowledge. The operator then makes one of three decisions: *accept* (promote to verified status), *reject* (move to a rejection archive with the

operator’s justification), or *defer* (return to quarantine for future review). No automated process may execute the accept decision. This constraint is architectural, not policy-based: the validation interface is the only code path through which an item’s status can transition to verified, and that code path requires an authenticated human action through the validation interface.

**Stage 5: Store.** Verified items are written to the sovereign temporal graph as nodes with full provenance metadata, temporal edges linking them to related knowledge, and an initial importance weight. The graph is *sovereign* in a specific technical sense: it is maintained under the exclusive governance authority of the data owner. No external platform, no upstream AI provider, and no automated optimization process may modify the verified status or provenance classification of stored items. Sovereignty here is an access control property enforced at the storage layer, not a deployment preference.

## 4.2 Tripartite Provenance Classification

Every item in the STG carries one of three provenance classifications, maintained as immutable metadata from the point of classification onward:

*Human-verified.* The item has passed through the validation stage (Stage 4) and received an explicit accept decision from an authenticated human operator. Human-verified items constitute the trusted substrate of the graph. They are the only items eligible for Core memory promotion (discussed in §5) and the only items that contribute to emergent topology formation.

*Source-system-inherited.* The item was imported from an external system with its original provenance metadata preserved. Source-system-inherited items must pass through the full pipeline, including human validation, before achieving verified status—preventing transitive trust propagation across system boundaries.

*AI-generated.* The item was produced or modified by a machine learning system—whether through extraction, summarization, inference, or any other computational process that involves a language model or statistical system. AI-generated items are permanently marked as such. This origin classification is retained in the provenance ledger even if the item is later endorsed through human validation. An AI-generated item may be *endorsed* by a human operator through the validation stage, which promotes it to human-verified status, but the provenance ledger retains the complete record that the item originated from an AI process.

## 4.3 The Negative-ML Constraint

The STG enforces what we term the *negative-ML constraint*: machine learning systems are structurally excluded from making final governance decisions on knowledge status. This is not a design heuristic or a best practice recommendation. It is an architectural invariant enforced through the system’s control flow.

Concretely: no code path exists through which a machine learning model’s output can directly change an item’s provenance classification from any non-verified state to human-verified. The validation interface (Stage 4) is the sole transition mechanism, and it requires authenticated human action. ML systems may assist in detection (Stage 2), flag items for review, compute risk assessments, and suggest classifications—but the governance decision itself is outside the ML system’s authority.

This constraint addresses a specific failure mode in existing architectures. Systems that use LLM-based importance scoring [9, 21] or automated memory consolidation [4, 21] allow the ML system to make decisions about what knowledge to retain, modify, or discard. When the ML system is also the source of potential contamination (through hallucination or confident error), allowing it to govern

its own outputs creates a self-validating loop: the system that introduced the error also decides whether the error should persist. The negative-ML constraint breaks this loop by ensuring that the governance authority and the potential contamination source are structurally distinct.

#### 4.4 Contamination Boundary Enforcement

The tripartite provenance classification enables a contamination boundary: a set of architectural rules governing which classes of items may interact with which. The central rule is:

*An item classified as AI-generated may not modify, overwrite, merge with, or be substituted for an item classified as human-verified without first passing through the quarantine and validation stages.*

This rule is enforced at the storage layer through write-path validation. When a write operation targets a node or edge in the graph, the storage layer checks the provenance classification of both the incoming item and the target item. If the incoming item is AI-generated and the target is human-verified, the write is rejected and the incoming item is routed to quarantine. This prevents the gradual contamination of verified knowledge through automated processes—a failure mode we term *silent overwrite*, in which an AI system’s updated summary or extracted fact replaces a human-verified record without any governance event.

The boundary is directional. Human-verified items may reference, annotate, or extend AI-generated items without restriction. The governance concern is specifically the integrity of the verified substrate: once knowledge has been human-verified, its modification requires human re-verification.

#### 4.5 Freeze and Revert as Temporal Governance Primitives

The STG provides two temporal governance primitives that operate on arbitrary subsets of the graph:

*Freeze.* A freeze operation designates a set of nodes and edges as immutable until an explicit unfreeze is executed by an authorized operator. Freeze protects critical knowledge during high-ingestion periods and provides a governance lock for knowledge validated through rigorous processes (clinical review, legal review, regulatory certification).

*Revert.* The STG maintains copy-on-write snapshots at configurable intervals and at every governance event. A revert restores any subset of the graph to a prior snapshot state, preserving the current state as a new snapshot before applying the restoration—no state is permanently lost. This addresses a governance scenario that flat memory systems handle poorly: discovering that a previously verified item was incorrect. In the STG, revert restores the pre-error state while preserving the error in the snapshot history for audit.

#### 4.6 Importance Weighting

Governance determines which items are eligible to participate in the graph as trusted knowledge; importance weighting governs how trusted items compete for retrieval and long-term prominence. Items stored in the STG carry an importance weight that evolves over time. We adopt the  $W(k, t)$  scoring function introduced in prior work on crystallization theory<sup>[12]</sup>, which computes importance as a function of initial verification strength, elapsed time subject to a temporal decay parameter  $\delta$ , access frequency, and cross-reference density with other verified items. We do not re-derive this function here; we use it as an established building block within the STG architecture. The key interaction between importance weighting and governance is that importance scores are computed only over human-verified items. AI-generated items in quarantine do not accrue importance, and their cross-reference density does not contribute to the importance of verified items they reference.

This prevents a contamination vector in which AI-generated items could inflate the importance of verified items they are designed to support, creating a form of importance laundering.

The sovereign temporal graph provides a substrate of trusted, governed knowledge. This substrate is a prerequisite for the architecture’s second component: emergent per-user topology depends on distinguishing foundational knowledge from noise. Without governance, clustering operates over an undifferentiated mass of verified and unverified items, and the resulting topology reflects contamination as much as genuine user concerns.

## 5. Emergent Per-User Topology

The previous section established a governed substrate of trusted knowledge. This section introduces the architecture’s second component: an emergent topology layer that organizes each user’s verified foundational memories into thematic clusters unique to that user. Where the sovereign temporal graph answers the question *what can we trust?*, the emergent topology layer answers the question *what does this user’s trusted knowledge look like when it self-organizes?*

The distinction from prior work is precise. Existing personalized memory systems organize user knowledge into predefined schemas—MemGPT/Letta uses developer-defined template blocks (persona, human) identical for every user [4]; Mem0 extracts facts into six fixed categories (personal information, preferences, events, temporal data, updates, assistant info) [21]; Generative Agents applies a uniform triple-scoring formula across a flat stream [13]. In each case, the organizational structure is imposed by the system designer, not derived from the user’s content. Separately, corpus-level systems like GraphRAG [10] demonstrate that emergent community detection can produce meaningful thematic clusters—but those clusters are properties of the corpus, identical for every user who queries it. STG-EUT produces clusters that are properties of the individual user: emergent, distinct, and dynamic.

### 5.1 Core Memory Designation

Not all verified knowledge participates in topology formation. The STG may contain thousands of human-verified items for a given user—factual extractions, conversational context, resolved queries—most of which are useful for retrieval but do not define the user’s identity or enduring concerns. A subset of these items, which we term *Core memories*, serve as the foundational nodes from which emergent topology is constructed.

Core memory designation is a human-verified promotion within the already-verified substrate. A user identifies a verified item as foundational—representing an enduring professional commitment, a deeply held value, a long-running project, or a defining personal experience—and confirms this designation through the same validation interface used for initial verification (§4, Stage 4). The confirmation is explicit and deliberate: the system may surface candidates for Core designation based on importance weight, access frequency, or cross-reference density, but the promotion itself requires authenticated human action. This is a second application of the negative-ML constraint: just as no automated process may verify knowledge, no automated process may designate knowledge as foundational.

Core memories receive two architectural privileges. First, they are assigned a permanence flag that exempts them from the temporal decay function applied to other verified items. While ordinary verified items gradually lose importance weight under the  $\delta$ -decay parameter of  $W(k, t)$ , Core memories maintain stable importance indefinitely. Second, they are treated as persistent high-priority context, available for default inclusion or preferential retrieval regardless of the current query. Where ordinary verified items must compete for retrieval through semantic similarity, Core

memories are eligible for inclusion in every computational context as baseline knowledge the system should always have available.

## 5.2 Per-User Knowledge Graph Construction

Core memories for a given user are organized into a per-user knowledge graph, denoted  $G_u = (V_u, E_u)$ , where  $V_u$  is the set of Core memory nodes and  $E_u$  is the set of edges representing relationships between them. Edges are computed along three dimensions:

*Semantic edges* connect Core memories whose content embeddings fall within a configurable similarity threshold. Two Core memories about patent filings in different technical domains, for example, would share a semantic edge reflecting their shared concern with intellectual property.

*Temporal edges* connect Core memories that were created, verified, or frequently co-accessed within proximate time windows. Temporal proximity often reflects thematic relatedness that semantic similarity alone does not capture—a user who verified several Core memories about a new business venture within the same month generates temporal edges among those items even if their content spans legal, financial, and technical domains.

*Thematic edges* connect Core memories that share extracted entities, referenced projects, or named concerns. These edges are derived from the structured metadata attached to each verified item during the governance pipeline, not from embedding similarity. A Core memory mentioning "Project Atlas" and another mentioning "Atlas team restructuring" share a thematic edge through the common entity, even if their semantic embeddings diverge.

The per-user knowledge graph is constructed incrementally. Each new Core memory designation triggers edge computation against all existing Core memories for that user, and the graph is updated in place. The graph is not rebuilt from scratch—it evolves as the user’s foundational knowledge evolves. Incremental graph maintenance keeps topology updates compatible with interactive memory systems, where full reconstruction on every Core memory change would be computationally impractical.

## 5.3 Community Detection and Emergent Cluster Formation

With the per-user knowledge graph constructed, the system applies unsupervised community detection to identify clusters of densely interconnected Core memories. We use a modularity-based community detection method (in our implementation, the Leiden algorithm<sup>[14]</sup>), which partitions graph nodes into communities that maximize within-community edge density relative to a random baseline. The algorithm requires no predefined number of clusters and no category labels—the number, size, and thematic identity of resulting clusters are emergent properties of the graph structure.

Each identified cluster is designated as an *Island*: a named thematic grouping of Core memories that represents one facet of the user’s identity or enduring concerns. The Island’s label is derived automatically from the most frequent entities, projects, and thematic keywords among its constituent Core memories. An Island is not a folder or a tag—it is a structural property of the user’s knowledge graph, defined by the density of connections among its members rather than by any external schema.

## 5.4 Why Different Users Produce Different Topologies

The same community detection algorithm, applied to different users’ knowledge graphs, produces structurally different Island configurations. This follows directly from the construction: because Core memories are designated individually and the edge structure reflects each user’s specific relationships, the input graph differs for every user. A user with Core memories spanning patent law,

healthcare, and music produces three distinct Islands; a user focused on clinical practice and education produces two. This differs from systems where all users share a universal schema: MemGPT’s fixed template blocks, Mem0’s six extraction categories, GraphRAG’s per-corpus communities.

### 5.5 Dynamic Lifecycle: Form, Merge, Weaken, Collapse

Islands are not static partitions. Because the per-user knowledge graph evolves as Core memories are added, modified, or removed, the community structure evolves correspondingly. We define four lifecycle transitions:

*Formation.* A new Island forms when a threshold number of Core memories cluster around a previously unrepresented theme—for example, a user beginning a new venture may verify foundational memories about its legal structure, technical architecture, and team, which cluster into a new Island once sufficient edges connect them.

*Merging.* Two Islands merge when cross-Island edge density exceeds within-Island boundaries—e.g., a user whose "patent law" and "healthcare" Islands accumulate cross-references through healthcare patent work. Merging is detected during periodic re-evaluation of the graph partition.

*Weakening.* An Island weakens when foundational Core memories are removed or demoted. Weakening manifests as reduced retrieval priority—the Island persists but ranks lower in query routing.

*Collapse.* An Island collapses when remaining Core memories fall below a minimum viability threshold in count or edge density. Collapsed Islands are archived in snapshot history; their Core memories remain in the graph and may participate in forming new Islands. Collapse represents the system recognizing that a facet of the user’s identity is no longer supported by sufficient foundational knowledge.

### 5.6 Cognitive Science Grounding

The architecture of emergent per-user topology is informed by, though not dependent on, three models from cognitive psychology. Conway’s Self-Memory System <sup>[15]</sup> describes autobiographical memory as organized into thematically coherent *lifetime periods* that emerge from experience rather than external categories—supporting the design choice that user-relevant clusters should self-organize rather than be predefined. Singer and Salovey’s self-defining memories <sup>[16]</sup> identify a class of vivid, repetitively recalled memories linked to enduring personal concerns that serve as identity anchors—supporting the Core memory mechanism as a small, disproportionately important subset. McClelland et al.’s Complementary Learning Systems <sup>[17]</sup> describe a two-tier architecture (rapid encoding plus gradual structure extraction) that produces meaningful organization over time—supporting the STG-EUT’s separation of fast ingestion from slow topology formation.

We cite these models as converging evidence that the emergent topology mechanism is cognitively grounded, not as formal theoretical foundations. The architecture stands on its systems properties; the cognitive science supports the claim that those properties align with how human memory self-organizes.

### 5.7 Distinction from Predefined Categorization and Corpus-Level Clustering

*Predefined categorization* (MemGPT/Letta, Mem0, ChatGPT) assigns every user the same developer-defined categories. A user whose concerns fall outside the schema has no mechanism for the system to create new categories. *Corpus-level clustering* (GraphRAG, RAPTOR) produces emergent categories, but as properties of the corpus—identical for all users—that do not adapt to individual identity or evolve with user behavior. STG-EUT differs on both axes: categories emerge from each

user’s verified foundational knowledge, and the input to clustering is itself a governed, user-specific structure. Emergent topology in STG-EUT is emergent *twice*: the graph emerges from verification decisions, and the clusters emerge from the graph.

Emergent per-user topology produces a set of thematic Islands for each user—clusters of foundational knowledge that represent the user’s enduring concerns and priorities. The next section shows how these Islands serve a dual function in the retrieval architecture: they narrow the search space to the most relevant thematic scope before fine-grained retrieval (a performance function), and they simultaneously constitute the system’s evolving structural model of the user’s enduring concerns (a perception function). The architectural insight is that these two functions are not independent features sharing a data structure—they are the same function viewed from two perspectives.

## 6. Identity-Aware Retrieval

The previous two sections presented the governance substrate (§ 4) and the emergent topology (§ 5). This section operationalizes both in the retrieval layer. Where flat memory systems answer a query by searching the entire memory store for the top- $k$  most similar items, STG-EUT first routes the query through the user’s Island topology to identify which thematic scope is most relevant, then performs fine-grained retrieval within that scope. The same Island structure that guides this routing also constitutes the system’s model of the user’s enduring concerns—a dual function that is the central claim of Contribution 4.

### 6.1 Island-Guided Query Routing

When a query arrives, the retrieval system executes a three-phase process:

**Phase 1: Island matching.** The system computes a relevance score between the query and each of the user’s active Islands. This score is derived from semantic similarity between the query and each Island’s Core memory content, thematic overlap between query entities and Island signatures, and recency of the user’s interaction with each Island. The output is a ranked list of Islands ordered by relevance to the current query.

**Phase 2: Scoped retrieval.** The system performs retrieval within the top-ranked Island’s scope. The scope includes not only the Island’s Core memories but also all non-Core verified items in the STG that are semantically or thematically associated with the Island—items that are within the Island’s thematic neighborhood but were not themselves designated as Core. This expansion is important: Core memories define the Island’s identity, but the Island’s retrieval scope includes the full breadth of relevant verified knowledge. Retrieval within scope uses standard vector similarity search, but over a substantially reduced candidate set compared to a full-store search.

**Phase 3: Expansion.** If scoped retrieval returns insufficient results—measured by a configurable minimum result count and minimum relevance threshold—the system expands to the next-ranked Island, then to the general verified store. Expansion is a fallback, not a default. In practice, queries that relate to a user’s established concerns are resolved within one or two Islands; queries on novel topics may require full-store search. The system tracks which queries required expansion, as persistent expansion failures for a given thematic area may indicate that the user’s Island topology does not yet cover that concern—a signal that may prompt Core memory candidacy suggestions.

### 6.2 Retrieval Narrowing

Island-guided routing is designed to produce a retrieval advantage over flat search, which we examine empirically in § 7. The mechanism is straightforward: by restricting the initial search to items within a single Island’s scope rather than the full verified store, the system reduces the

candidate set over which vector similarity must be computed. For a user with  $n$  total verified items distributed across  $k$  Islands of approximately equal size, scoped retrieval searches approximately  $n/k$  items rather than  $n$  items in the common case where the query matches a single Island.

The relevance advantage is less obvious but equally important. Items within an Island’s scope share thematic coherence because they are associated—through semantic, temporal, or thematic edges—with Core memories that the user has verified as foundational to a specific concern. This means the candidate set for scoped retrieval is not a random partition of the full store but a thematically concentrated subset. The probability that any individual retrieved item is relevant to the query is higher within a correctly matched Island than within the full store, because the Island’s scope is defined by the same thematic criteria that make items relevant to queries about that theme.

The combined effect is a retrieval system that is both faster (smaller candidate set) and more precise (thematically concentrated candidates) than flat search, without requiring the user to manually specify a search scope or select from a list of predefined categories.

### 6.3 The Identity/Perception Function

The Island topology that guides retrieval simultaneously serves a second function: it constitutes the system’s structural model of the user’s enduring concerns and priorities. Each Island represents a facet of the user’s identity as expressed through their verified foundational knowledge. The complete set of Islands—their labels, their relative sizes, their interconnections, and their lifecycle states—forms a map of who the user is from the system’s perspective.

This model is not a static profile. Because Islands form, merge, weaken, and collapse as Core memories change (§5.5), the identity model evolves with the user. A user who begins a new professional venture will see a new Island form as they verify foundational memories about the venture. A user who leaves a field will see the corresponding Island weaken and eventually collapse as they revoke Core status from memories that are no longer central to their concerns. The system’s model of the user is not a snapshot—it is a living topology that tracks the user’s evolving identity through the dynamics of their foundational knowledge.

This perception function has concrete operational consequences. When the system surfaces Core memory candidates for the user’s review, it can identify gaps in the topology—thematic areas where the user has significant verified knowledge but no Core memories, suggesting that the area may be underrepresented in the identity model. When the system presents retrieval results, it can annotate them with Island provenance—indicating which facet of the user’s identity each result relates to.

### 6.4 Why Retrieval and Perception Are Architecturally Inseparable

An Island narrows retrieval scope because its Core memories are densely connected around a coherent theme. That dense, coherent thematic structure is *exactly* what makes the Island an interpretable facet of the user’s identity. If Core memories were randomly connected, the Island would be useless for both retrieval (random scope does not improve relevance) and perception (a random cluster is not an identity facet). Conversely, a user profile not organized into the same structures used for retrieval would drift from the retrieval layer—the system would “know” who the user is but would not use that knowledge to improve retrieval.

The inseparability is a consequence of the architecture. The same graph, the same community detection, and the same lifecycle dynamics produce both retrieval scopes and the identity model. This contrasts with Mem0, which maintains independent profile and vector store structures, and GraphRAG, which builds community structures for retrieval without per-user identity modeling.

## 6.5 Distinction from Corpus-Level Retrieval and Static Hierarchies

*GraphRAG DRIFT search* <sup>[10]</sup> is the closest existing mechanism to Island-guided routing, performing community-level matching before local retrieval using Leiden-family community detection. The distinction is threefold: GraphRAG communities are per-corpus (identical for all users), unfiltered (no governance on indexed content), and single-function (retrieval only). STG-EUT applies community detection per-user, over governed foundational memories, for both retrieval and perception.

*RAPTOR* <sup>[11]</sup> constructs hierarchical retrieval trees through recursive abstractive processing. Like GraphRAG, it operates per-corpus with a static hierarchy that does not vary across users or evolve without reindexing. STG-EUT’s Islands are dynamic, per-user, and dual-function.

*Flat vector retrieval* (ChatGPT, MemGPT, Mem0) searches the full store without intermediate routing. STG-EUT replaces this with a two-phase process: identify the relevant thematic scope, then search within it.

The preceding sections have presented the STG-EUT architecture in full: sovereign temporal governance (§4), emergent per-user topology (§5), and identity-aware retrieval (§6). The next section demonstrates these architectural properties through a prototype implementation and case study, showing that the system produces divergent topologies for different user profiles, that Island-scoped retrieval can improve on flat search in the prototype case study, and that the topology responds to Core memory perturbation as the lifecycle model predicts.

## 7. Prototype and Case Study

To evaluate the architectural properties described in §4–§6, we implemented a prototype of the STG-EUT pipeline and conducted a case study demonstrating topology emergence, retrieval behavior, and lifecycle dynamics. We frame this section as a proof-of-concept demonstration of the architecture’s structural properties, not as a general benchmark. The prototype validates that the pipeline functions end-to-end and that its outputs exhibit the behaviors the architecture predicts. A full multi-user longitudinal evaluation is future work.

### 7.1 Implementation Overview

The prototype implements the STG-EUT pipeline as a Python-based processing system operating over exported conversational history from a commercial AI assistant. The prototype is a structurally faithful but simplified surrogate of the full architecture: governance is simulated through rule-based classification and human review of output files, and topology is approximated through category-based organization rather than graph-based community detection. The case-study pipeline used in this section approximates the full governance loop, even though the underlying system already implements confirmation-gated lifecycle control, freeze/revert operations, and recall interfaces at the service level. The system comprises three components:

*Ingestion and parsing.* A conversation parser reads the exported JSON/JSONL format, extracts human-assistant message pairs, and tags each pair with available temporal metadata. In the prototype, 2,425 conversation pairs were extracted from approximately 18 months of interaction history (20MB of source data). Trivial exchanges (greetings, single-word confirmations) were filtered, and content-level deduplication removed 4 near-duplicate pairs based on normalized content hashing of the first 500 characters.

*Classification and governance simulation.* Each pair passes through a keyword-based classifier assigning it to one or more of 28 thematic categories organized into three governance tiers: *Crystallize* (14 categories—patents, architecture decisions, legal structures, identity-relevant history),

*Hot* (11 categories—active operations, current projects, recent technical work), and *Archive* (3 categories—general coding, historical projects, uncategorized). An exclusion filter removes content from discontinued projects (828 items excluded). This classification simulates the governance pipeline described in § 4: in production, the keyword classifier would be replaced by ML-assisted triage followed by human validation, but the tier logic would be identical.

*Knowledge file generation.* Classified items are aggregated by category into Markdown files (capped at 100,000 characters), organized by date with source attribution. The pipeline produced 106 knowledge files totaling 8.7 million characters. A manifest provides a verification checklist for human review.

The prototype is grounded in a deployed service-backed memory system. Beyond the analysis pipeline, the implementation includes control surfaces for memory retention, recall, human confirmation, lifecycle consolidation, and state-governance operations (freeze/revert) exposed through a Model Context Protocol (MCP) interface with eight registered tools, including a designated confirmation endpoint serving as the sole crystallization path—operationalizing the negative-ML constraint (§ 4.3).

## 7.2 Controlled Comparative Topology Demonstration

To demonstrate that the same pipeline produces different topological structures from different interaction histories, we conducted a controlled comparative demonstration using two user profiles processed through identical classification and aggregation stages.

*User A* is a technology entrepreneur and healthcare technologist whose interaction history spans patent drafting, infrastructure architecture, peptide commerce operations, music production, and personal introspection. Processing User A’s 2,425 conversation pairs produced 24 active categories with the following Crystallize-tier distribution: technical planning (181 entries), healthcare patent work (93 entries), personal reflection (239 entries), IP licensing and deals (85 entries), regulatory compliance (119 entries), Azure infrastructure (60 entries), MCP tool integrations (107 entries), and crystallization theory (90 entries), among others. The topology reflects a user whose foundational concerns span multiple professional domains and include substantial personal identity content.

*User B* is a simulated profile constructed by filtering User A’s history to retain only healthcare-related and technical architecture content, then supplementing with synthetic conversational pairs reflecting a clinical researcher’s interaction pattern. This comparison is intended to demonstrate sensitivity of the category-level topology to input-history composition, not to establish generalizable cross-user results. Processing this filtered history through the same pipeline produced 9 active categories with a markedly different distribution: clinical workflows (312 entries), research methodology (187 entries), FHIR/interoperability (94 entries), grant writing (45 entries), and patient data governance (78 entries), among others. Personal reflection, music production, patent law, and commerce categories—prominent in User A’s topology—are absent entirely.

The comparison demonstrates the central claim of § 5.4: the same classification algorithm, applied to different interaction histories, produces structurally different category configurations. User A’s topology has 24 categories spanning 5 professional domains plus personal identity content. User B’s topology has 9 categories concentrated in a single professional domain. The resulting organizational structure is primarily a property of the interaction history being processed rather than a fixed property of the system.

## 7.3 Retrieval Comparison

We compared scoped retrieval (within the matched category) against flat retrieval (across all verified content) on a set of 50 test queries spanning User A’s thematic concerns.

*Scope reduction.* For queries matching a single category, scoped retrieval searched an average of 4.2% of the total verified item count (the matched category’s entries divided by total entries). The median scope reduction was 23×—a query about patent architecture searched 93 items rather than the full 2,174 verified items. Queries matching multiple categories searched the union of matched scopes, averaging 11.3% of total items.

*Relevance.* We evaluated relevance through manual inspection of the top-5 retrieved items for each query under both scoped and flat conditions. For queries with a clear thematic focus (42 of 50), scoped retrieval returned more thematically coherent results: all 5 items related to the query’s domain in 38 cases under scoped retrieval, compared to 29 cases under flat retrieval. The remaining 8 queries—those spanning multiple domains or addressing novel topics—showed no relevance difference, as scoped retrieval expanded to flat search when within-scope results were insufficient.

These results are illustrative rather than statistically generalizable. They should be read as architectural plausibility evidence rather than as a controlled retrieval benchmark. The prototype uses keyword-based classification rather than the full governance pipeline, and the test queries were constructed by the system developer. A controlled evaluation with independent assessors and diverse user populations is needed to establish generalizable retrieval advantages.

## 7.4 Topology Perturbation

To test the lifecycle dynamics described in § 5.5, we simulated Core memory removal by reclassifying all items in User A’s “healthcare patent” category from Crystallize to Archive tier (removing 93 entries from the foundational substrate) and re-running the classification pipeline.

The perturbation produced three observable effects. First, the healthcare patent category collapsed—it no longer appeared in the active topology. Second, 14 items previously classified as healthcare patent content were reclassified into adjacent categories (regulatory compliance, IP licensing) based on secondary keyword matches, demonstrating that removal of a foundational category redistributes some content to neighboring categories rather than losing it entirely. Third, the topology contracted from 24 to 23 active categories, with the total Crystallize-tier entry count decreasing from 1,368 to 1,289. The contraction was localized: categories unrelated to healthcare patents (music, personal reflection, infrastructure) were unaffected.

This perturbation test is a simplified analog of the Island collapse mechanism described in § 5.5. In a full implementation with community detection over a per-user knowledge graph, the effects would propagate through edge removal rather than keyword reclassification. The prototype demonstrates the qualitative behavior—that removing foundational items causes localized topological contraction—while the precise dynamics of graph-based collapse remain to be validated in a production implementation.

## 7.5 What the Prototype Demonstrates and What It Does Not

The prototype demonstrates five properties: (1) the governance substrate is operational as a deployed service with confirmation-gated crystallization, freeze/revert, and MCP-exposed lifecycle control; (2) the end-to-end pipeline processes 2,425 conversation pairs into 106 structured knowledge files in under 60 seconds; (3) category-level topology diverges across user profiles; (4) scoped retrieval reduces search space 23× while maintaining relevance; (5) removing foundational content causes localized topological contraction.

The distinction between implemented and approximated components is important. The governance substrate—confirmation-only crystallization, freeze/revert, tiered lifecycle, recall/consolidation—is implemented as a deployed service. The emergent topology layer—graph-native community

detection, dynamic Island lifecycle from edge density, live interactive identity-aware retrieval—is approximated through keyword-based categories. Multi-user longitudinal evaluation has not been conducted. The governance architecture is validated at the service level; the emergent topology architecture is validated at the structural-plausibility level.

The prototype confirms that the STG-EUT architecture produces the structural behaviors it is designed to produce: governed ingestion, user-specific topology, scoped retrieval, and lifecycle dynamics. The next section examines the architecture’s costs, constraints, and the distance between the prototype and a production system.

## 8. Discussion and Limitations

The STG-EUT architecture addresses a genuine gap in the AI memory landscape: the absence of systems that combine contamination-aware governance with emergent per-user topology. The prototype validates that the architecture’s core structural properties—governed ingestion, user-specific categorization, scoped retrieval, and lifecycle dynamics—are realizable. This section examines the architecture’s costs, constraints, and the distance between the prototype and a production system.

### 8.1 Human Verification Bottleneck

The architecture’s strongest claim—that human-gated validation prevents the self-reinforcing contamination loop—is also its most expensive constraint. Every item that achieves verified status must pass through human review. Every Core memory promotion requires a second human confirmation. In a system processing hundreds of conversational interactions per week, the verification queue grows faster than most users can review.

The architecture partially mitigates this through triage: the detection stage (§4, Stage 2) surfaces high-importance candidates first, and the importance-weighting function concentrates verification effort on items with the highest expected impact. The power-law distribution of knowledge importance—where a small fraction of items accounts for a disproportionate share of retrieval value—suggests that prioritizing the highest-impact minority of items may capture much of the governance benefit. Nevertheless, the bottleneck is real. Users with low engagement tolerance for verification workflows may find the governance overhead unacceptable.

This is a design tradeoff, not a deficiency. Systems that automate verification (allowing the LLM to decide what to store and what to forget) achieve higher throughput at the cost of the governance guarantees that motivate the architecture. The negative-ML constraint is the architectural decision that sovereignty requires human cost. Whether that cost is acceptable depends on the value of the knowledge being governed—in domains where contamination has high consequences (clinical, legal, financial), the cost is justified; in casual use cases, it may not be.

### 8.2 Prototype Limitations Versus Full Architecture

Three specific gaps separate the prototype from a production system. First, governance is simulated through keyword classification rather than a live five-stage pipeline with real-time quarantine and validation. Second, topology is approximated through category-based organization rather than graph community detection—the categories demonstrate divergence but not the emergent, label-free clustering that is the architecture’s central topological claim. Third, the dual-function mechanism has not been exercised through a live interactive system with real-time Island-matched query routing. These gaps are acknowledged; a production implementation is required to validate performance under operational conditions.

### 8.3 Scalability and Maintenance Costs

STG-EUT introduces costs that flat systems avoid: provenance metadata, snapshot history, contamination boundary enforcement, incremental graph updates, and periodic community detection. For a single user, these are manageable—the prototype runs in under 60 seconds on commodity hardware. For multi-tenant deployment with thousands of independent knowledge graphs, capacity planning for aggregate graph maintenance and snapshot storage is an open question we have not evaluated.

### 8.4 Cold-Start and Sparse-Topology Problems

The emergent topology mechanism depends on a sufficient density of Core memories to produce meaningful clusters. A new user with few or no Core memories has no Islands, no scoped retrieval, and no identity model. The system degrades gracefully to flat retrieval in this case, but the architecture’s distinctive properties are unavailable until the user has verified enough foundational memories to populate at least two or three Islands.

A related problem is sparse topology: a user who designates many Core memories that are thematically diffuse—with weak semantic, temporal, and thematic edges among them—may produce a per-user graph too sparse for community detection to identify coherent clusters. The result would be either a single undifferentiated Island (offering no retrieval advantage over flat search) or many small Islands with only one or two Core memories each (too fragmented to serve as meaningful identity facets). The minimum viable density for useful Island formation is an empirical question that the prototype does not answer.

### 8.5 Governance–Convenience Tradeoff

The architecture’s sovereignty model places governance authority with the data owner. This is a deliberate design choice with a cost: the system cannot automatically learn from user interactions without human confirmation at the verification and Core memory designation stages. Systems that automate these decisions—Mem0’s LLM-driven extraction, ChatGPT’s automatic memory insertion—provide a more convenient user experience at the cost of the governance guarantees STG-EUT is designed to provide.

This tradeoff is not resolvable within the architecture’s design constraints. Relaxing the negative-ML constraint to allow automated verification would reintroduce the self-validating loop that the constraint exists to prevent. Allowing automated Core memory designation would undermine the claim that Islands reflect the user’s deliberate identity rather than the system’s extraction heuristics. The architecture accepts lower automation in exchange for higher governance integrity. Whether this tradeoff is appropriate depends on the deployment context and the stakes associated with contaminated knowledge.

### 8.6 What the Paper Does and Does Not Establish

This paper establishes four things. First, the Dual Contamination Problem is a real systems failure mode that existing memory architectures do not explicitly address in compound form. Second, the STG governance architecture provides an architectural mechanism—not merely a policy recommendation—for preventing the self-reinforcing contamination loop through tripartite provenance, quarantine, human-gated validation, and contamination boundary enforcement. Third, the architecture plausibly supports emergent per-user topology, and the prototype demonstrates category-level divergence consistent with that claim. Fourth, the dual-function property—where the same emergent structure serves both retrieval and identity modeling—is a consequence of the architecture rather than a feature bolted on afterward.

This paper does not establish general performance superiority over existing memory systems. It does not provide a controlled multi-user benchmark. It does not validate the full graph-based community detection mechanism in production. It does not measure the identity perception function’s effect on downstream task quality. And it does not resolve the governance–convenience tradeoff—it names it.

The contributions are architectural. The architecture is validated at the structural level. Performance validation, user studies, and longitudinal evaluation across diverse populations are necessary future work before the claims can be extended from architectural feasibility to operational superiority.

## 9. Conclusion

Current AI memory systems store knowledge in flat, ungoverned structures where all items carry equal weight, contamination sources are indistinguishable, and organizational categories—when they exist—are predefined by system designers rather than derived from the user’s own concerns. This paper presented an alternative: the Sovereign Temporal Graph with Emergent User Topology, an architecture that unifies contamination-aware governance with emergent per-user knowledge topology. The sovereign temporal graph provides a governed substrate through tripartite provenance, quarantine, a human-gated validation boundary, and freeze/revert state control. The emergent topology layer organizes each user’s human-verified foundational memories into thematic clusters that form, merge, and collapse as the user’s concerns evolve. The same cluster structure that narrows retrieval scope also constitutes the system’s evolving model of the user’s enduring priorities—a dual function that is a consequence of the architecture rather than a feature added to it.

The contributions are architectural. The paper formalizes the Dual Contamination Problem, presents a governance mechanism that prevents the self-reinforcing contamination loop, demonstrates category-level divergence consistent with the per-user topology claim, and shows that retrieval optimization and identity perception are architecturally inseparable in the proposed design. A prototype validates these properties at the structural level; performance evaluation across diverse users and operational conditions remains future work. The broader question the paper raises—whether AI memory systems should organize knowledge into structures that reflect each user’s identity rather than a universal schema—is one the field will need to answer as persistent memory becomes standard infrastructure for AI applications.

## References

- [1] OpenAI. ChatGPT Memory. 2024. <https://openai.com/index/memory-and-new-controls-for-chatgpt/>
- [2] Zep AI. Graphiti: A temporal knowledge graph architecture for agent memory. arXiv:2501.13956, 2025.
- [3] J. Tappolet and A. Bernstein. Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. Proc. ESWC, 2009.
- [4] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv:2310.08560, 2023.
- [5] L. Moreau and P. Missier. PROV-DM: The PROV data model. W3C Recommendation, 2013.
- [6] R. Liang, F. Zhao, and J. Zhang. Blockchain-based data provenance: A survey. IEEE Access, 2020.
- [7] DoltHub. Dolt: Git for data. <https://github.com/dolthub/dolt>, 2024.

- [8] Y. Xu, Y. Liang, et al. A-MEM: Agentic memory for LLM agents. arXiv:2502.12110, 2025. (NeurIPS 2025)
- [9] W. Zhong, L. Guo, Q. Gao, and Y. Wang. MemoryBank: Enhancing large language models with long-term memory. Proc. AAAI, 2024.
- [10] D. Edge, H. Trinh, N. Cheng, et al. From local to global: A graph RAG approach to query-focused summarization. Microsoft Research, arXiv:2404.16130, 2024.
- [11] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. Proc. ICLR, 2024.
- [12] R. Valentine.. Crystallization theory: A mathematical framework for knowledge validation in AI systems. #63/953,509, 2025.
- [13] J. S. Park, J. C. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. Proc. UIST, 2023.
- [14] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: Guaranteeing well-connected communities. Scientific Reports, 9(1):5233, 2019.
- [15] M. A. Conway and C. W. Pleydell-Pearce. The construction of autobiographical memories in the self-memory system. Psychological Review, 107(2):261-288, 2000.
- [16] J. A. Singer and P. Salovey. The Remembered Self: Emotion and Memory in Personality. Free Press, 1993.
- [17] J. L. McClelland, B. L. McNaughton, and R. C. O’Reilly. Why there are complementary learning systems in the hippocampus and neocortex. Psychological Review, 102(3):419-457, 1995.
- [18] J. Gutierrez-Basulto, J. C. Jung, and T. Schneider. Lightweight description logics and branching time: A formal investigation. Proc. KR, 2014.
- [19] N. Gottschalk and B. Demuth. EventKG: A multilingual event-centric temporal knowledge graph. Proc. ESWC, 2018.
- [20] J. Fernandez, M. Arias, and M. Polleres. HDT-FoQ: Compressed full-text indexing for efficient SPARQL queries over large RDF datasets with provenance. Knowledge and Information Systems, 2019.
- [21] D. Chhikara, P. Khant, et al. Mem0: Building production-ready AI agents with scalable long-term memory. arXiv:2504.19413, 2025.

**Acknowledgments.** This work was developed independently. The author thanks the open-source AI research community for tools and frameworks that enabled rapid prototyping and specification development.

**Conflicts of Interest.** The author has filed patent applications related to the Crystallization Theory framework (#63/953,509) and the KEOS architecture (USPTO #63/962,609).

**Code and Data Availability.** A reference implementation of the STG-EUT governance service (Memibrium) is available at <https://github.com/rvalen1123/memibrium>. The knowledge extraction pipeline used in the case study is available upon request.