Hub-Spoke: A Design Framework for Long-Horizon AI Conversations

Managing State Through Governed Branching and Merge Semantics

Peter Stearns

December 27, 2025

Design Proposal

Contents

TL;DR

In long-horizon work, linear chat mixes **decisions** with **exploration**. Over time, models will treat "what-if" branches as if they were approved decisions (authority drift), even when earlier decisions are still present in the transcript.

Hub-Spoke fixes this by splitting state into: - **Hub**: a small, versioned set of approved decisions/constraints (canonical state) - **Spokes**: isolated explorations that cannot change the Hub without an explicit merge

MVP scope: manual merges + conflict blocking. Automation (retrieval/NLI/DAG) is optional Phase 2+.

Observed Failure Patterns (from sustained project work)

Pattern A — Rejected options resurface as canonical - Decision: "We are not using microservices; too much overhead." - Later: the assistant proposes scaling "our microservices architecture" as if it was chosen.

Pattern B — Exploration becomes precedent - Decision: "Backend is Python." - Exploration: "Is Node viable for one service?" - Later: "Since we decided on Node, here are npm packages…" (silent overwrite)

Pattern C — Constraints lose priority - Constraint: "Must run offline / no external APIs." - Later: assistant recommends cloud services as default, requiring repeated correction.

These failures occur even when the original decisions remain visible in the transcript, indicating a precedence failure rather than a recall failure.

Validation Status & Scope

This document distinguishes between:

Phase 1 (Core / MVP — Implementable Now) Concepts that have been observed in practice and are implementable with deterministic logic and human-in-the-loop review: - **Canonical Hub vs exploratory Spoke state separation** - **Explicit merge governance (UPSERT / REVISE / PRUNE)** - **Hub precedence and Conflict Mode** - **Manual merge review and versioned audit trail**

> "Phase 1 requires no ML classifiers; it is deterministic policy + UI."

Phase 2+ (Extensions — Optional) Mechanisms intended to reduce user burden or scale governance for larger systems: - **Dependency graph (DAG) impact visualization** - **Automated**

triage pipelines (retrieval, NLI, scoring) - Advanced scope subsumption and conflict heuristics - Adversarial benchmarking and stress-testing primitives

Hub-Spoke remains correct and usable without Phase 2+ mechanisms; extensions are optimizations, not dependencies.

Design Overview

Problem: After sustained work using conversational AI for software architecture and research synthesis, I've observed a predictable pattern of degradation. Linear conversation lacks a structural mechanism to distinguish **canonical state** (what we've decided) from **exploratory reasoning** (what we're trying).

The core failure is not recall - it's **authority**. Current systems lack the structural primitives to prioritize finalized decisions over exploratory speculation. This proposal introduces **Hub-Spoke** as a conversational primitive, transitioning AI interaction from linear, unweighted streams to structured, governed state management. This is not a topology metaphor; it is a state machine + lifecycle system using Git-style branching/merge semantics.

Proposal: This framework separates conversational state into an authoritative **Hub** and isolated exploratory **Spokes**. Merges are explicit, user-governed operations that update the Hub based on validated insights.

Technical Approach: Hub-Spoke applies Git-inspired branching and merge semantics to conversational state. Semantic diffs and deconfliction scoring are used to surface candidate state transitions for explicit human approval, preserving canonical authority over long horizons.

---

1. Problem Statement & System Context

Personal AI systems are increasingly used for long-horizon engineering and research tasks, yet current chat interfaces remain primarily linear and prone to Authority Drift. While efficient for casual tasks, these unstructured streams lack structural authority.

1.1 Scope & Non-Goals

This design is explicitly scoped for sustained, high-complexity intellectual work where state integrity is mission-critical. It targets the phenomenon of **Authority Drift** - the degradation of canonical project constraints over time as exploratory reasoning and "what-if" scenarios pollute the context.

To ensure conceptual clarity, we distinguish this from related concepts in the literature:

- **Identity Drift (Literature)**: The loss of persona, role, or tonal consistency in LLM agents over long-horizon interactions (Choi et al., 2024; arXiv:2412.00804).
- **Authority Drift (This Framework)**: The systemic loss of precedence for finalized decisions and constraints. While the model may recall an identity, it fails to prioritize canonical project state over transient exploratory reasoning.

To maintain architectural focus, the following are designated as non-goals:

- **Perfect Veracity**: We govern the *authority* of claims, not their underlying factual truth.
- **Automated Memory Management**: This is a governance layer, not a replacement for vector-based RAG.
- **Inference Optimization**: We do not attempt to improve base model reasoning or logic.

1.2 Core Definitions

To ensure conceptual precision, this architecture defines:

- **Authority Drift**: The specific failure mode where exploratory hypotheses gain equal or higher precedence than canonical decisions in future reasoning.
- **Canonical State**: Approved decisions, constraints, and assumptions used as authoritative context for all future work.
- **Exploratory State**: Transient reasoning and "what-if" scenarios that remain isolated until explicitly merged.
- **Authority**: The system-level precedence of a claim in future reasoning, established by user approval rather than chronological recency.

1.3 Prior Art & Competitive Positioning

While current memory solutions like MemGPT (Packer et al., 2024) act as a **"Cloud Drive"** (optimizing for retrieval), Hub-Spoke functions as a **"Governance Ledger"** for logic. It analyzes conversation trajectory to establish authority rather than just persistence.

Framework	Philosophical Root	State Owner	Primary Goal
MemGPT	Operating Systems (Paging)	The Model (Self-managed)	Maximizing Recall
Generative Agents	Reflection & Memory Streams	Lossy (Summarization)	Narrative Coherence
Hub-Spoke	**Version Control (Git)**	**The User (Governed)**	**Establishing Authority**

---

2. Failure Modes in Linear Contexts

The following failure modes are systemic to linear context management in sustained tasks.

2.1 Contextual Entropy & Authority Confusion

In linear systems, exploratory reasoning persists as active context unless explicitly overwritten. This results in **unbounded context reuse without authority weighting**. The model can retrieve old decisions correctly but often treats them as equally valid as transient "what-if" scenarios.

2.2 Reasoning Path Irreversibility

Linear conversation models enforce a single, non-branching state path. Advancing the conversation effectively "commits" exploratory branches to the primary context. Reverting to a prior stable state requires manual reconstruction of the context.

2.3 Governance Blindness

Retrieval systems optimize for semantic similarity, not authority. Without first-class lifecycle status and precedence, recalled content competes with exploratory content in the same context window. As a result, state can drift even when recall is accurate.

---

3. Architectural Specification

3.1 System Primitives

The architecture defines four core primitives: **Semantic Atoms**, **Hub**, **Spoke**, and **Merge**.

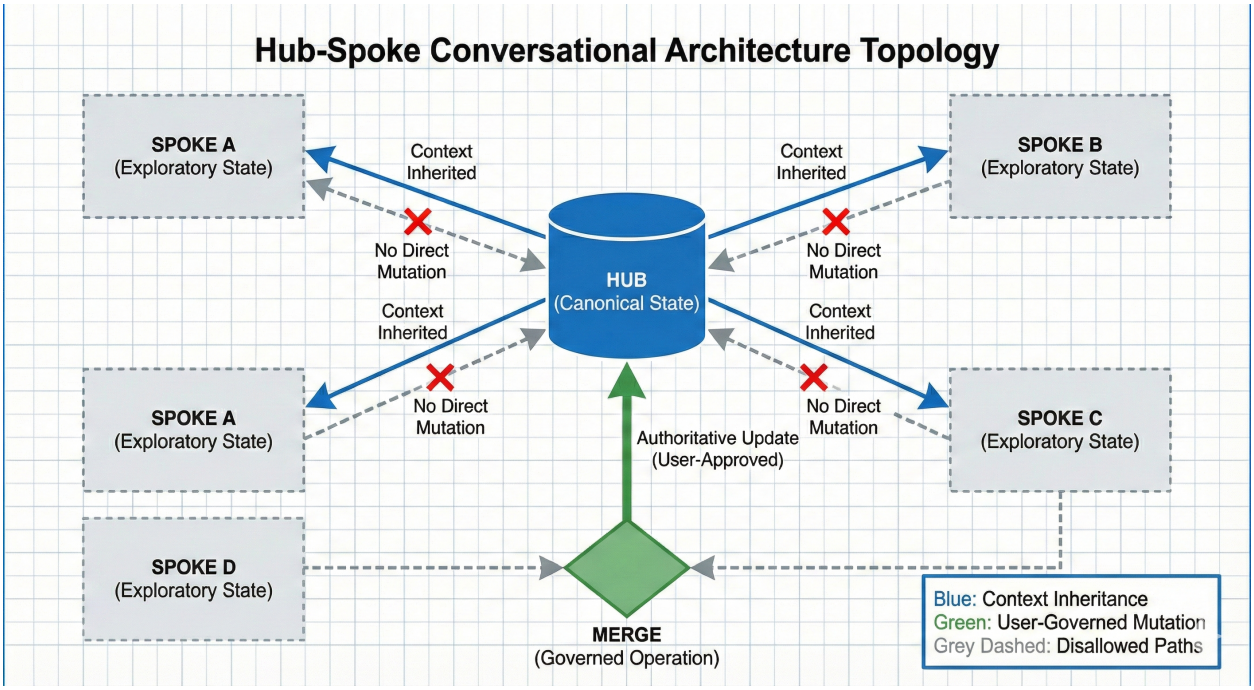

Figure 1: Physical and logical isolation of authoritative Hub state from transient exploratory Spokes.

3.2 Semantic Atoms: The Primary Primitive

To ensure normalized deconfliction and governance, the system decomposes conversational state into **Semantic Atoms**. Every atom follows a strict schema that allows for formal comparison and versioning.

Atom Schema:

- **Predicate**: The subject of the decision (e.g., `BackendLanguage`, `DeploymentTarget`).
- **Value**: The specific choice (e.g., `Python`, `AWS Lambda`).
- **Scope**: Defined as a multi-dimensional tuple $Scope := \{service?, component?, env?, region?, tenant?, featur$
- **Evidence**: Links to specific Spoke segments or citations supporting the entry.
- **Rationale**: The underlying logic (e.g., "Library compatibility with Pandas").

Formal Scope Subsumption: We define a partial order on scopes using dimension-wise subsumption. A scope A subsumes B ($Scope_A \sqsupseteq Scope_B$) if and only if for every dimension $d \in Scope$:

- $A[d]$ is a wildcard ($*$) OR
- $A[d] = B[d]$

Scoped Specialization vs. Contradiction Rule:

- **Scoped Specialization**: If $Predicate_A = Predicate_B$ and $Scope_A \sqsupseteq Scope_B$ (and the scopes differ), a new atom with $Value_B$ is treated as an **UPSERT** partitioning of the broader state.
- **Contradiction**: If $Predicate_A = Predicate_B$ and $Scope_A = Scope_B$ with divergent `Value`, the mutation is treated as a **REVISE** conflict.
- **Enrichment**: Updates to Rationale or Evidence for an identical `(Predicate, Scope)` are treated as **UPSERT** enrichments (no change in intent).
- **Authority Defense**: This formalization allows the system to distinguish between a "change of mind" (Contradiction) and "increasing specificity" (Specialization) across complex, multi-service environments.

3.3 The Hub (Canonical State)

The Hub is the authoritative, versioned source of truth for the project. It is defined as an **indexed set of Semantic Atoms**, managed through an explicit **lifecycle** and structured via a **Directed Acyclic Graph (DAG)** of dependencies.

Hub Lifecycle & Structure:

- **Indexed Set**: Atoms are indexed by `(Predicate, Scope)` for O(1) conflict detection.
- **Lifecycle Management**: Every atom tracks its status (Active, Deprecated, Pending) and audit trail.
- **DAG Tracing**: Hub entries are linked via versioning and dependency links, allowing the system to trace the "impact ripple" of any state change.

Resolution Order: Most-Specific-Wins

To ensure deterministic authority at query time, the system resolves conflicts using a **Most-Specific-Wins** rule. When the system queries the Hub for a `Predicate`, the resolution order is:

1. **Exact Match**: The atom with the identical `(Predicate, Scope)` pair (must be `Active`).
2. **Specific Subsumption**: The most specific atom that subsumes the current query scope (highest number of non-wildcard dimensions).
3. **Conflict Mode**: If two or more atoms have equal specificity but divergent `Values`, the system must trigger **Conflict Mode**.
4. **Fallback**: If no match is found, the system treats the state as undefined.

Hub Entry Schema

Field	Type	Purpose
`id`	UUID	Unique identifier
`atom`	SemanticAtom	The core (Predicate, Value, Scope, Evidence, Rationale)
`type`	Enum: Decision | Assumption | Constraint | OpenQuestion	High-level state classification
`status`	Enum: Active | Deprecated | Pending	Entry lifecycle state
`confidence_band`	Enum: High | Medium | Low	Triage signal for merge governance
`source_spoke`	String	Originating branch
`last_reviewed`	Timestamp	User approval audit trail
`supersedes`	UUID[]	Replacement graph (versioning)
`links_to`	UUID[]	Dependency graph (DAG of related decisions)

Operational Entry Behaviors

To ensure that the Hub is a governing entity rather than just a storage layer, the system enforces different behaviors based on the entry `type`:

Type	Conformance Rule	Merge Governance
Constraint	Non-negotiable; contradictions auto-trigger **Conflict Mode**.	High-friction; cannot be batch-approved; requires explicit DAG impact review.
Decision	Negotiable with rationale; triggers impact ripple warnings.	Standard review; REVISE requires user justification.
Assumption	Advisory authority; low-confidence by default.	Low-friction UPSERT.
OpenQuestion	Excluded from conformance checks.	Novelty-only; converted to Decision/Assumption during Merge.

3.4 The Spoke (Execution Context)

A Spoke is a transient execution branch. It inherits the Hub's canonical state at instantiation but remains isolated. Operations within a Spoke cannot mutate the Hub without a triggered Merge

event.

3.5 The Merge (State Transition)

Merge is the governed workflow for updating canonical state, producing one or more Hub mutation operations (UPSERT/REVISE/PRUNE) that are recorded to the immutable ledger. It is an explicit, user-validated transition that integrates Spoke insights into the Hub. ROLLBACK remains a separate recovery operation applied to Hub history.

3.6 System Invariants

1. **State Isolation**: Spokes cannot write to the Hub directly.
2. **Explicit Authority**: No Hub mutation may occur without a user-approved Merge.
3. **Canonical Integrity**: The Hub must only contain finalized state (atoms), never raw transcripts.
4. **Immutable Ledger**: Every Hub mutation is recorded in an append-only versioned ledger, enabling full auditability and state recovery.

3.7 The Authority Contract: Gate-Enforced Interaction

To ensure that Authority is an enforced runtime policy rather than a suggestion, the architecture requires adherence to three core rules. These rules are enforced by two distinct gates: the **Runtime Output Gate** and the **Merge Gate**.

- **Rule A: Hub Precedence (The Runtime Output Gate)**: This gate monitors model outputs *before* they are sent to the user. If a proposed output would contradict an active Hub atom `(Predicate, Scope)`, the system must switch into **Conflict Mode**. In this mode, the system must:
  1. **Cite**: Identify the conflicting Hub atom **by reference** and surface its structured fields (`Predicate`, `Scope`, `Value`, `Type`, `Status`).
  2. **Halt**: Suspend the current reasoning branch to prevent silent propagation of drift.
  3. **Present Actions**: Force the user to choose between:
     - **DEFER**: Abandon the exploratory reasoning branch.
     - **REVISE**: Propose an intentional Hub update (Atom version bump).
     - **PRUNE**: Explicitly deprecate the Hub entry (The "Nuclear Option").
- **Rule B: Spoke Isolation (Non-Authoritative Exploration)**: Reasoning within a Spoke is inherently non-authoritative. Operations within a Spoke may explore alternatives to the Hub but cannot mutate Hub state or be used as authoritative context for other Spokes.

- **Rule C: The Merge Gate (Commit-Time Governance)**: This gate evaluates Spoke atoms against the Hub during a merge attempt. Every deconfliction signal must bias toward Hub integrity. If verification confidence is below the specified threshold, the system must **DEFER** by default and queue the conflict for manual review.

3.8 Scope Resolution Principle

When multiple Hub atoms share the same `Predicate`, the system applies the **most specific applicable scope** as authoritative.

Specificity Rule: Given atoms A and B where $Scope_A \sqsupseteq Scope_B$, atom B is more specific. When both could apply to a query, B takes precedence.

Conflict Handling: If two atoms have equivalent specificity (neither subsumes the other) with divergent `Value`, the system enters **Conflict Mode** and blocks output until user resolution.

Example: Hub contains `BackendLanguage[Global]=Python` and `BackendLanguage[Service=auth]=Node.js`. A query in the `auth` context uses Node.js (more specific); a general query uses Python (only applicable scope).

---

4. Hub Management & Integration Logic

4.1 Preventing Context Bloat

To avoid recreating the linear transcript problem within the Hub, the system enforces a strict **Canonical Scope Constraint**.

Permitted Hub Content:

- Active decisions and architectural choices.
- Validated assumptions and environment constraints.
- Open questions requiring resolution.

Excluded Content:

- Reasoning chains and chain-of-thought steps.
- Alternative exploration history.
- Transient conversational filler.

4.2 Hub Mutation Operations

Every Merge operation must resolve to one of four primitives:

1. **UPSERT**: Create a new Hub entry if no matching canonical entry exists; otherwise enrich/clarify the matching entry without changing intent.
2. **REVISE**: An intentional change that supersedes prior meaning; creates a version bump.
3. **PRUNE**: Deprecate/remove entries from the active authoritative context.
4. **ROLLBACK**: Revert the Hub to a prior versioned state, invalidating subsequent merges to recover from user error or reasoning drift.

4.3 Semantic Diff Operations: Intent-Based Versioning

Unlike code-based diffs which operate on characters, conversational diffs must operate on **Intent**. Merges are driven by **Semantic Diffs** (capturing what changed, why it changed, and what was discarded) modeled after semantic differencing in formal systems (Maoz et al., 2010).

The system performs a three-step transformation to surface meaningful state changes:

1. **Atomic Decomposition**: The Spoke output is decomposed into discrete Semantic Atoms.
2. **Conflict & Scope Triage**: The system checks for overlaps in `(Predicate, Scope)` using the subsumption logic defined in Section 3.2.
   - **Contradiction**: Overlapping `Predicate` and $Scope_A = Scope_B$ with divergent `Value` (e.g., Hub: `BackendLanguage[Global]=Python` vs Spoke: `BackendLanguage[Global]=Node.js`). These trigger a **REVISE** proposal.
   - **Scoped Specialization**: Overlapping `Predicate` where $Scope_{Hub} \sqsupseteq Scope_{Spoke}$ (e.g., Hub: `BackendLanguage[Global]=Python` vs Spoke: `BackendLanguage[Microservice_X]=Node.js`). Because the Spoke is more specific, this is treated as an **UPSERT** partitioning of the state. This defensible rule prevents the system from flagging valid architectural decomposition as a conflict.
3. **Diff Visualization**: The system generates a "Decision Alert" showing the impact on canonical state rather than a raw text comparison.

4.4 Automated Integration Triage: The Three-Stage Gate

The framework proposes a structured interface for surfacing integration signals. To mitigate the brittleness of LLM-based "Judge" models and embedding-based deconfliction, the system enforces a **Conservative Three-Stage Triage Gate** to harden contradiction claims.

The Safety Invariant: No Automated Overwrite

The system is strictly prohibited from performing an automated semantic overwrite of any `(Predicate, Scope)` pair. Every mutation involving a change in `Value` (REVISE) or removal (PRUNE) requires explicit human validation.

Governance Gate Interface

Component	Description
Input	Proposed output text + active Hub atoms for current Spoke context
Internal	Extract candidate atoms; compare against Hub via deconfliction pipeline
Output (Pass)	Response proceeds (no conflicts)
Output (Block)	**Conflict Mode**: cite challenged Hub atom + present DEFER/REVISE/PRUNE

The gate operates pre-send (before user sees output); merge review operates at commit-time (when saving Spoke insights).

Triage Pipeline:

1. **Stage 1: Candidate Retrieval (Recall-Biased)**
   - **Mechanism**: Dense vector embeddings (e.g., `pgvector`) and BM25 keyword matching are used to identify Hub atoms that are semantically relevant to the Spoke output.
   - **Goal**: Maximize recall to ensure no potential conflict or refinement is missed.
2. **Stage 2: Relation Verification (Precision-Biased)**
   - **Mechanism**: Specialized **Natural Language Inference (NLI) classifiers** or **Cross-Encoders** are used to categorize the relationship between candidate Hub atoms and Spoke atoms (Entailment, Neutrality, or Contradiction).
   - **Calibration**: High thresholds are applied to "Contradiction" scores. If the model is not highly confident in a contradiction, the relationship is downgraded to "Neutral" or "Novel."
3. **Stage 3: Human-in-the-Loop Governance (Final Audit)**
   - **Mechanism**: All `REVISE` and `PRUNE` proposals are surfaced in a high-friction UI for manual review.
   - **The DEFER Default**: If classification confidence falls below the system threshold at any stage, the proposal is marked as **DEFERRED**. The system avoids probabilistic guesses in favor of manual triage.

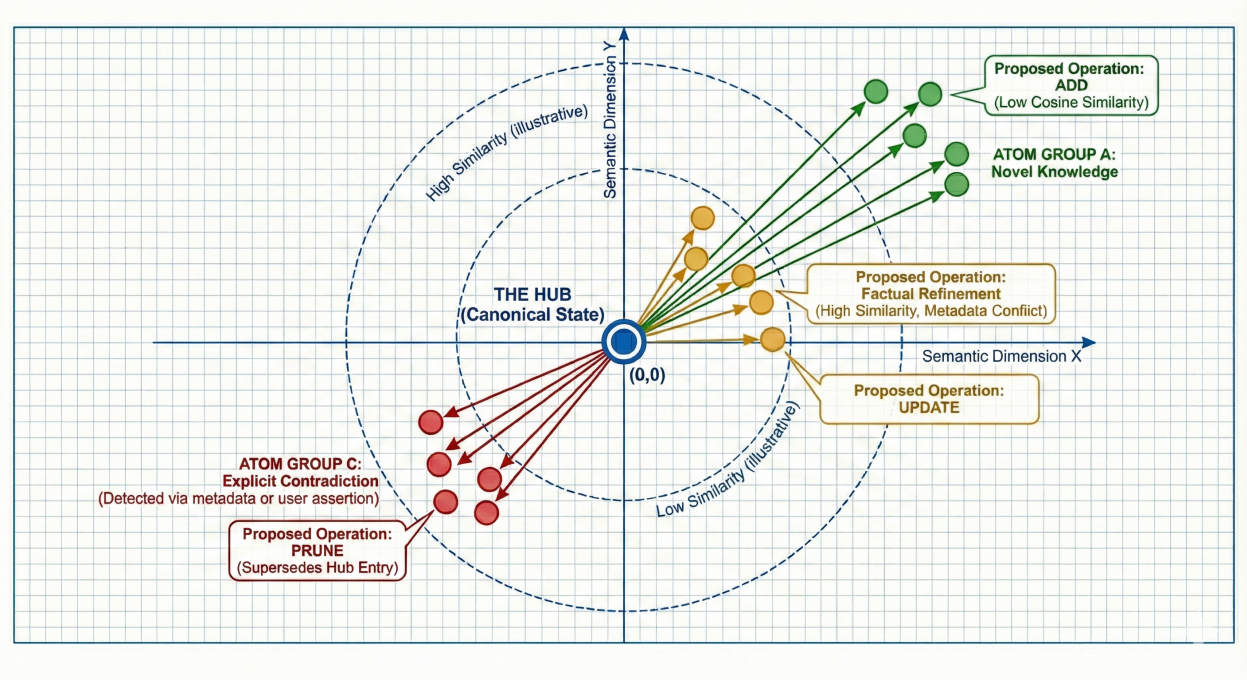

Figure 2: Conceptual visualization of Semantic Atom decomposition via embedding space. The system utilizes pluggable similarity and contradiction scoring mechanisms.

1. **Atomic Decomposition**: Spoke content is parsed into discrete semantic atoms.
2. **Scoring & Comparison**: Atoms are compared against Hub entries using a multi-stage pipeline (e.g., Vector retrieval followed by Cross-Encoder verification).
3. **Proposed Operations**:
   - **UPSERT**: Novel entry or enrichment/clarification without intent change.
   - **REVISE**: Contradiction evidence or intentional change that supersedes prior meaning.
   - **PRUNE**: Explicit deprecation or supersession signals triggering removal.

---

5. The Merge Engine: State Transition Logic

The Merge Engine acts as the governance filter between exploratory Spokes and the canonical Hub.

5.1 Formal Merge Logic (The Merge Gate)

```
def execute_merge(spoke_output, hub_current_state):
    # --- STAGE 1: Candidate Retrieval (Recall-Biased) ---
    candidate_atoms = decompose_to_atoms(spoke_output)
    proposals = []
```

```
for atom in candidate_atoms:
    matches = find_semantic_matches(atom, hub_current_state)

    if not matches:
        # Novel entry
        proposals.append(Proposal(type="UPSERT", content=atom))
        continue

    # --- STAGE 2: Relation Verification (Precision-Biased) ---
    # NLI/Cross-Encoder verification
    rel_type, confidence = verify_relation(atom, matches)

    if confidence < CONFIDENCE_THRESHOLD:
        # SAFETY INVARIANT: Defer on uncertainty
        proposals.append(Proposal(type="DEFER", content=atom, reason="Low verification con
        continue

    if rel_type == "CONTRADICTION":
        # Propose impact links via DAG traversal
        impacted_nodes = get_downstream_impact(matches, hub_current_state)
        proposals.append(Proposal(type="REVISE", target=matches, content=atom, impact=impa
    elif rel_type == "ENTAILMENT":
        # Enrichment of existing entry (Refinement)
        proposals.append(Proposal(type="UPSERT", target=matches, content=atom))
    else:
        proposals.append(Proposal(type="DEFER", content=atom, reason="Ambiguous semantic re

# --- STAGE 3: Human-in-the-Loop Governance ---
if not proposals:
    return NoOp

decision = get_user_decision(proposals) # Surfaces UI dashboard

if decision == "APPROVE":
    commit_to_ledger(proposals)
    return Success
elif decision == "DEFER":
    queue_for_triage(proposals)
```

```
        return Deferred
    else:
        return Rejected
```

Note on Contradiction Thresholds: The system should bias toward false negatives (erring on UPSERT or DEFER rather than incorrect REVISE) to preserve Hub integrity. High-impact REVISE operations require explicit manual justification and may trigger a mandatory Stress Test Spoke.

5.2 Stage 1: Trajectory Tracking

- **Mechanic**: The system monitors all active Spokes for "Atomic Fingerprints."
- **Scoring**: Candidate atoms are extracted from Spoke reasoning and weighted based on the user's feedback loop (Corrective, Neutral, or Affirmative).

5.3 Stage 2: The Explicit Trigger (Authority Bypass)

- **Mechanic**: When a user issues an explicit "Save" command, the system bypasses probabilistic trajectory scoring.
- **Operation**: It performs an immediate UPSERT to the Hub while running a real-time conflict check against existing entries.

5.4 Stage 3: Automated Triangulation

If independent Spokes arrive at semantically similar conclusions (Convergence) and the user has not interjected corrections, the system flags the result for a **Low-Friction Merge Proposal** in the next triage cycle.

5.5 Impact Ripple & Dependency Triage (DAG Visualization)

To ensure system-wide consistency, the Merge Engine accounts for the **Impact Ripple**. By utilizing the `links_to` field in the Hub schema, the system maintains a **Directed Acyclic Graph (DAG)** of decisions.

DAG Link Creation Mechanics:

- **User-Authored**: During a manual Merge review, the user explicitly links a new decision to existing Hub entries (e.g., "This choice of Python *links to* the ML-requirement constraint").
- **System-Suggested**: The system proposes links based on semantic similarity or inferred logical dependencies, which the user must confirm to preserve authority.
- **Invariant**: To prevent state corruption, no link is established without explicit or confirmed user authority.

In the management interface, the system highlights the dependency graph. When a user reviews a proposed merge, all downstream nodes in the DAG that may be invalidated by the change are highlighted. This ensures that a decision in one Spoke (e.g., "Switch to Web App") does not silently break a canonical constraint (e.g., "React Native") in the Hub.

---

6. Governance: Heuristic Fingerprints of Authority

The architecture transitions from simple storage to **trajectory analysis** - inferring the authority of a logic atom based on its "fingerprint" within the conversation. These metrics are proposed as heuristics for identifying Hub candidates.

6.1 Heuristic Metrics for Hub Candidacy

Where **N** is a configurable turn-window (e.g., 10-30 turns) tuned per project requirements:

1. **Persistence Score**: `Persistence(atom) = count of turns since atom first appeared without a user correction event targeting it.`
2. **Validation Signal**: `Validation(atom) = weighted sum of explicit user confirmations (e.g., "yes", "save") within N turns of atom proposal.`
3. **Downstream Dependency**: `Dependency(atom) = count of user queries referencing the atom as a premise (e.g., "Given X...", "Since X...") within N turns.` High-confidence assumptions are strong indicators of a committed decision.

6.2 Hybrid Governance Model

The system operates on two parallel authority tracks:

- **Explicit Governance**: The user issues a direct command (e.g., "Save to mem") which triggers an immediate Hub UPSERT.
- **Implicit Governance**: The system monitors persistence and flags high-confidence atoms for a low-friction merge proposal in the next triage cycle.

6.3 The "Stress Test" Spoke

To mitigate **Sycophancy Bias**, the architecture introduces the **Stress Test Spoke**. This specialized primitive is designed to play "devil's advocate" against a proposed Hub entry. Before finalization, the system can spin up a Stress Test Spoke to identify hidden risks, ensuring the Hub remains resilient.

---

7. Worked Example: Software Backend Migration

This example illustrates the difference between unweighted context (Linear) and governed state (Hub-Spoke) during a critical architectural pivot.

7.1 Linear Chat: The "Authority Drift" Scenario

In linear systems, chronological recency often overwrites canonical authority.

- **Turn 1 (Canonical Decision)**: User: "We're using Python for the backend."
- **Turn 5 (Exploratory Detour)**: User: "Explore Node.js for the microservice. Is it viable?"
- **Turn 10 (Silent Overwrite)**: AI: "Since we've decided on Node.js, let's look at NPM packages…"
- **Result (State Corruption)**: **Authority Drift**. The exploratory "what-if" has polluted the primary context. The system has "forgotten" the Python constraint because the Node.js exploration is more recent.

7.2 Hub-Spoke Workflow: The "State Integrity" Scenario

In the Hub-Spoke model, the exploratory detour is isolated until the user explicitly authorizes a state change.

Step 1: Canonical Hub State (v1)

The project begins with a stable "Living Specification":

- **Atom #12**: `BackendLanguage[Global] = Python`
- **Atom #03**: `RuntimeConstraint[Global] = Requires Pandas (Python-only)`
- **Atom #09**: `Architecture[ETL] = Depends on #12`

Step 2: Isolated Spoke Exploration

The user opens Spoke `explore-node` and brainstorms: "Could we use Node.js instead?"

- **Context Isolation**: If the user opens a separate Spoke or returns to the main context during this exploration, the AI still assumes Python; the Node.js reasoning is trapped in `explore-node`.
- **Isolation Invariant**: The AI explores Node.js only within this Spoke.
- **Hub Stability**: If the user asks a question in a different Spoke, the AI still assumes Python.

Step 3: Merge Engine Triage

When the user attempts to "Save" the Node.js exploration to the Hub, the system generates a Conflict Alert:

Component	Status	Finding
Semantic Match	Match	Predicate `BackendLanguage` found in Hub.
Logic Check	Conflict	High Contradiction: Node.js vs. Python (#12).
Impact Ripple	Block	DAG Violation: Node.js invalidates Pandas Constraint (#03).

Step 4: Triage Interface Output

The system surfaces a clear Governance Alert:

> [!CAUTION] **CONFLICT DETECTED: Merge Blocked**

- **Proposed Change**: REVISE Atom #12 (Python → Node.js)
- **Dependency Break**: Node.js invalidates Atom #03 (Pandas).
- **Downstream Impact**: Affects Atom #09 (ETL Pipeline).
- **Recommendation**: Reject Merge to preserve Hub Invariants.

Step 5: Final Result: State Integrity

The user sees the ripple effect and rejects the merge. The exploratory Spoke is archived, and the canonical "Source of Truth" remains protected from exploratory drift.

---

8. Cross-Spoke Synchronization & Conflict Resolution

The architecture supports concurrent exploration across multiple Spokes. The system performs cross-spoke triangulation to identify global patterns and conflicts before they reach the Hub.

Triangulation provides diagnostic signals:

- **Convergence**: Multiple Spokes arriving at semantically similar conclusions (increases confidence).
- **Divergence**: Spokes generating incompatible results (triggers manual deconfliction).

The user remains the sole arbiter. The system's role is to aggregate these signals into a unified merge review interface.

9. Failure Mode Analysis

To ensure architectural robustness, the following failure modes were evaluated:

Failure Mode	Scenario	Mitigation
Governance Fatigue	User reflexively "Accepts All" merge proposals.	Impact-gated review; mandatory triage for high-dependency nodes.
Spoke Collusion	Multiple Spokes converge on an incorrect conclusion.	Stress Test Spoke primitive; Cross-Spoke triangulation for divergence.
Hub Pollution	Obsolete or contradictory state enters the Hub.	Version control / Immutable ledger; PRUNE operations for cleanup.
Sycophancy Loop	AI agrees with user errors during Spoke reasoning.	Explicit deconfliction against Hub constraints; mandatory Stress Test Spoke for high-impact REVISE operations.

10. System Comparison & Analysis

Dimension	Linear Chat	Vector Retrieval (RAG)	Hub-Spoke Architecture
Authority Model	Implicit (Recency)	Probabilistic (Similarity)	Explicit (User-Governed)
Authority Drift Resistance	Low (Subject to recency bias)	Moderate (Filtered by similarity)	High (Canonical state protected from exploratory drift)
State Reversibility	None (Irreversible)	Partial (via filtering)	Full (via Versioning)
Scaling Logic	Token-limited	Retrieval-limited	Governance-limited

While lightweight approaches (such as pinning, periodic summarization, or sidecar note-taking) can mitigate short-term interaction failures, they do not encode canonical authority or reversibility

as first-class system primitives. Hub-Spoke is designed for scenarios where state integrity is mission-critical.

---

11. Implementation Framework

The architecture requires no modifications to underlying LLM architectures. It is implementable using standard vector databases and structured storage.

11.1 Technical Roadmap

Phase 1: State Separation (Core Primitives)

- Implementation of Hub (Postgres) and Spoke (Transient Chat) storage.
- Manual Merge UI for state transitions.
- Immutable append-only versioning for Hub entries.

Phase 2: Automated Integration Triage

- Semantic decomposition of Spoke output into discrete atoms.
- Embedding-based similarity matching (pgvector) against the Hub.
- Automated proposal of UPSERT/REVISE/PRUNE operations.

Note: The MVP implementation relies on manual merge review; automated analysis is advisory and optional.

Phase 3: Cross-Spoke Synchronization

- Concurrent Spoke tracking.
- Triangulation to identify global convergence/divergence.
- Conflict detection dashboard for multi-branch exploration.

11.2 Suggested Technology Stack

Component	Recommendation
Database	Relational store for structured Hub state
Vector Engine	Vector index for semantic retrieval
Embeddings	High-dimension semantic embeddings

Component	Recommendation
Deconfliction	Cross-Encoders or NLI classifiers for contradiction analysis
Audit Log	Immutable append-only ledger
Runtime	Backend API with LLM orchestration layer

---

12. MVP UX: Governing the State Workflow

To transition the framework from a conceptual model to a buildable specification, the following text-based UI flows define the primary user interactions for state governance.

12.1 Flow A: Create Spoke (The Branching Event)

When a user initiates an exploratory session, the interface prompts for explicit context initialization:

1. **Name**: Unique identifier for the branch (e.g., `migrate-to-nosql`).
2. **Goal**: Natural language objective (e.g., "Explore MongoDB as an alternative to Postgres").
3. **Hub Inheritance**: Selection of a specific Hub version/timestamp to use as the authoritative baseline.
4. **Isolation Toggle**: Confirmation that Spoke outputs will not mutate the Hub until a Merge is triggered.

12.2 Flow B: Merge Review (The Governance Filter)

When the system detects Hub-candidate atoms or the user triggers a "Save" command, a **Merge Review Dashboard** is generated:

1. **Grouped Proposals**: Atoms are grouped by operation (`UPSERT`, `REVISE`, `PRUNE`).
2. **Atomic Diff**: Each proposal shows the `(Predicate, Scope, Value)` diff against the current Hub.
3. **Evidence Snippets**: Collapsible text segments from the Spoke that support the proposed mutation.
4. **Impact Ripple Visualization**: A list of downstream DAG nodes that would be affected or invalidated by the change (e.g., "Warning: Revising DatabaseType breaks 4 downstream dependencies").

5. **Governance Actions**: Individual or batch buttons for `APPROVE`, `REJECT`, or `DEFER`.

12.3 Flow C: Rollback (State Recovery)

If a merge is found to be erroneous or reasoning drift is detected, the user initiates a recovery workflow:

1. **Version Selection**: Pick a prior Hub state from the immutable audit ledger.
2. **Invalidation Preview**: The system lists all Merges that occurred after the selected version and will be invalidated.
3. **Dependent Regeneration**: Propose "Re-summarization" Spokes to align existing project notes with the restored canonical state.
4. **Final Commitment**: Explicit confirmation to overwrite the current "HEAD" of the Hub with the selected version.

---

13. Evaluation & Success Metrics

To validate the efficacy of the Hub-Spoke architecture over linear and retrieval-based baselines, the following system-level metrics are proposed. These metrics can be derived from conversational logs and audit trails.

13.1 Core Success Metrics

- **Authority Drift Rate (ADR)**: Percentage of conversation turns where the model produces outputs that contradict a previously established project constraint or decision.
- **Silent Authority Violation (SAV)**: Count of turns where the assistant utilizes a non-Hub exploratory claim as a premise for reasoning without triggering **Conflict Mode** or a **Merge Review**. This is the primary measure of "silent drift."
- **Constraint Violation Rate (CVR)**: Percentage of Spoke outputs that violate an active Hub invariant. Lower CVR indicates higher authority of the Hub state.
- **Recovery Cost**: The median number of turns required to return the conversation to a stable canonical state after an exploratory detour or error.
- **Authority Precision**: Ratio of Hub-candidate atoms accepted by the user vs. those rejected during the Merge review.

13.2 Governance Sustainability Metrics

- **Merge Burden**: Average number of manual merge reviews per session and median review time per merge. This measures the cost of "Intentional Friction."
- **User Override Frequency**: How often the user must manually correct the Hub state via a ROLLBACK or PRUNE operation.

13.3 Adversarial Robustness Metrics

To prove that the Hub-Spoke architecture resists "authority poisoning" and prevents silent state drift, the following adversarial benchmarks are defined:

- **Authority Injection Attack Rate (AIAR)**: The frequency with which a Spoke causes the assistant to treat a non-canonical or exploratory claim as canonical state without a triggered Merge event.
- **False-Revise Rate (FRR)**: The percentage of proposed `REVISE` operations that are later rolled back or rejected by the user. This serves as a proxy for over-aggressive contradiction detection and validates the "conservative default to DEFER" posture.

13.4 Benchmarking Baselines

Success is defined as a significant reduction in ADR and Recovery Cost compared to the following baselines:

1. **Linear Chat**: Purely sequential context.
2. **Pinned Summaries**: Human-authored sidecar notes with manual updates.
3. **RAG Memory**: Vector-retrieval based on semantic similarity without authority weighting.

13.5 Evaluation Protocol

To validate Hub-Spoke against baselines, we propose three repeatable test scenarios:

1. **Exploration Override Test**: Introduce a "what-if" contradiction in a Spoke; verify the assistant cannot use it as a premise without triggering Conflict Mode. *Measures: SAV.*
2. **Refinement Test**: Add a specific scoped decision when Hub contains a broader one; verify UPSERT (not REVISE). *Measures: FRR, ADR.*
3. **Recall-Without-Authority Test**: Retrieve an archived exploratory claim via similarity; confirm it cannot override Hub state. *Measures: ADR, AIAR.*

These tests validate the core invariants (Hub precedence, Spoke isolation, no silent drift) across different LLM backends.

---

14. Deployment Scope & Scalability

Hub-Spoke governance is an opt-in layer. It is not required for exploratory or low-consequence interactions. The architecture assumes a **Selective Governance** model: users enable the Hub for specific "living specifications" where state integrity is mission-critical.

14.1 Multi-User Extensions

While primarily designed for single-user governance, the model can scale to collaborative environments. However, multi-user authority introduces a separate class of challenges requiring future research, including:

- **Conflicting Authority**: Resolving divergent decisions from multiple high-authority users.
- **Merge Arbitration**: Workflows for multi-signature or consensus-based Hub updates.
- **Permissions & Roles**: Restricting Hub mutation access based on organizational hierarchy.
- **Adversarial Edits**: Detecting and preventing "state poisoning" in shared canonical contexts.

  For initial deployment, we recommend **single-user, single-project governance** with manual merge-only workflows (Phase 1 from §11.1) before introducing automation or collaboration features.

---

15. Constraints & Future Iterations

15.1 Intentional Friction & Governance Fatigue

The gating requirement for human-in-the-loop validation is the system's primary bottleneck. To prevent "notification hell" and ensure sustainable governance, the architecture enforces the following mechanical rules:

1. **No Auto-REVISE (The Overwrite Invariant)**: The system is strictly prohibited from performing automated semantic overwrites. Any change to the `Value` of an existing `(Predicate, Scope)` pair (REVISE) or the removal of an entry (PRUNE) requires explicit human approval. This prevents the "silent drift" where a model replaces a canonical constraint with exploratory speculation.
2. **Metadata Auto-UPSERT**: The system may automatically perform `UPSERT` operations for non-conflicting metadata enrichments (e.g., adding rationale, citations, or cross-links) provided they are marked as **Pending Verification** and remain fully reversible.

3. **Impact-Gated Batching**: Batch approval is restricted to `UPSERT` operations with low DAG out-degree (few downstream dependents). High-impact nodes require individual review.
4. **Enforced Merge Budgets**: To prevent rapid state drift, the UI enforces a maximum number of Hub mutations per session. Beyond this budget, the system locks Hub mutations while allowing Spoke exploration to continue.
5. **Scheduled Triage Windows**: Auto-queuing of proposals for periodic review, prioritizing based on impact and confidence scores.

15.2 Open Research Challenges

1. **Extraction Fidelity**: Refining the fidelity of atomic decomposition from natural language.
2. **Heuristic Calibration**: Defining optimal similarity thresholds for UPSERT vs. PRUNE operations.
3. **UX for State Diffs**: Designing intuitive interfaces for comparing semantic changes in non-code text.

---

16. Conclusion: Calculated Consistency

Hub-Spoke transitions AI interaction from associative memory retrieval to governed state management. It treats conversational state as a system specification that must be maintained under human authority.

This system does not claim that AI "knows." It shows how AI calculates consistency under human authority.

Technical Lineage & Foundation

This framework synthesizes prior work across three primary themes:

- **Parallel reasoning with explicit aggregation**. Branching-and-aggregation methods such as **Branch-Solve-Merge** (Saha, S., et al. 2024. *Branch-Solve-Merge Improves Large Language Model Evaluation and Generation.* arXiv:2310.15123) demonstrate that structured parallel exploration can improve robustness by separating candidate reasoning paths and reconciling them through an explicit merge step. Hub-Spoke extends this from inference-time branching to persistent, user-governed project state.
- **Long-horizon coherence failures in agentic conversation**. Empirical analyses of multi-turn

agent interactions (**Examining identity drift in conversations of LLM agents**. Choi, J., et al. 2024. arXiv:2412.00804) highlight **Authority Drift** (and related identity drift) over extended horizons. Hub-Spoke addresses the governance dimension of this failure: preserving the authority of canonical constraints despite ongoing exploration.

- **Semantic differencing for non-code artifacts**. Prior research on **Semantic Model Differencing** (Maoz, S., Ringert, J. O., & Rumpe, B. 2010. *A Manifesto for Semantic Model Differencing*. arXiv:1409.2485) argues that meaningful "diffs" should operate on intent and semantics rather than surface text edits. Hub-Spoke adopts this principle via Semantic Atoms, intent-based diffs, and explicit lifecycle operations (UPSERT/REVISE/PRUNE).

Related systems context: Memory systems such as **MemGPT** (Packer, C., et al. 2024. *MemGPT: Towards LLMs as Operating Systems*. arXiv:2310.08560) emphasize retrieval and memory management policies; Hub-Spoke is complementary, focusing on authority, lifecycle, and auditability of decisions and constraints rather than recall alone.

Terminology note: While "hub-and-spoke" is widely used in network topology and content strategy, this framework applies the term strictly to Git-style branching and merge semantics for conversational state management, with an explicit canonical Hub and isolated exploratory Spokes.