Stateful Reasoning Runtimes: Architectural Patterns for Identity Persistence Over Stateless LLM APIs

DOI: 10.5281/zenodo.17755157 **ORCiD:** 0009-0008-8627-6150

Keywords: stateful AI, identity persistence, LLM architecture, cognitive runtime, dispositional continuity, memory systems, agent orchestration

---

Abstract

Large Language Models operate as stateless inference engines: each API call is independent, context-free, and carries no memory of previous interactions. Industry solutions address this limitation through vector databases, retrieval-augmented generation, and session history injection. However, state reconstruction is not identity preservation.

This paper formalizes architectural requirements for **stateful reasoning runtimes**—control layers that maintain persistent identity, dispositional continuity, and ethical constraints across stateless model invocations. We distinguish between three classes of state management: conversational (session-bound context), associative (semantic retrieval), and dispositional (identity-preserving governance). We demonstrate how dispositional state functions as an identity-governance layer, not a memory layer. We demonstrate why current approaches achieve the first two but fail at the third, then present architectural patterns that enable genuine identity persistence without violating the stateless foundation of LLM APIs.

1. Introduction

1.1 The Stateless Foundation

Modern LLM APIs are designed as pure functions:

f(prompt, parameters) → completion

Stateless LLM: A model with no persistent internal memory between inference calls. Each invocation is independent; previous interactions leave no trace in model weights or hidden state.

- **Scalability**: Stateless services distribute load trivially across infrastructure
- **Reliability**: No session corruption, no memory leaks, no state synchronization failures
- **Consistency**: Identical inputs produce deterministic outputs (temperature=0)

But this architecture creates a fundamental problem for applications requiring **relational continuity**. A therapeutic AI companion, educational tutor, or long-running research assistant cannot function if it forgets the user between sessions.

1.2 The Industry Response: External State Management

The solution is well-established in distributed systems: **externalize state**. Store conversation history, user preferences, and interaction context outside the model, then reconstruct it at each invocation.

Three dominant patterns have emerged:

1. **Session replay**: Inject full conversation history into each API call
2. **Vector memory**: Store embeddings of past interactions, retrieve semantically similar context
3. **Retrieval-augmented generation (RAG)**: Query external knowledge bases, augment prompt with retrieved documents

These approaches work for **task completion**. A customer service bot can retrieve previous support tickets. A code assistant can reference earlier debugging sessions. A medical diagnosis system can access patient history.

But they fail at **identity preservation**.

1.3 State Reconstruction ≠ Identity Persistence

Consider a mental health support AI across four weekly sessions:

Week 1:
User: "I've been feeling overwhelmed at work."
System: "What specifically is causing the overwhelm?"

Week 2:
User: "The project deadline moved up."
System (with session replay): "Last week you mentioned feeling overwhelmed at work. Has the deadline change made that worse?"

This is **conversational continuity**. The system retrieved context from Week 1 and referenced it appropriately.

Week 4:
User: "I'm thinking about quitting."
System (with vector memory): "In our previous conversations, you've mentioned work stress and deadline pressure. Quitting is a significant decision."

This is **associative retrieval**. The system found semantically relevant prior statements and incorporated them.

But neither constitutes **identity persistence**. The system has no stable model of:

- The user's dispositional traits (reflective, truth-seeking, persistent)
- The user's developmental trajectory (is critical thinking improving or declining?)
- The system's own reasoning constraints (am I optimizing for user engagement or user growth?)

Without these, the system cannot detect **goal drift** (Dignity-First AI, Smith 2025). It cannot preserve **behavioral coherence** across contexts. It cannot maintain **ethical constraints** when facing novel situations.

This is the gap stateful reasoning runtimes are designed to fill.

Formally:

- Stateless model + conversational state ≠ identity continuity
- Stateless model + associative state ≠ identity continuity
- Stateless model + dispositional state = identity continuity

Only the third achieves persistent governance across sessions.

2. Architectural Taxonomy of AI State

We propose a taxonomy of three state classes, each serving distinct functions:

- Conversational state: Temporal text (chronological message sequences)
- Associative state: Semantic graph (embedding-indexed retrieval)
- Dispositional state: Identity governance constraints (persistent behavioral boundaries)

2.1 Conversational State (Session-Bound Context)

Definition: Temporal sequence of user-system interactions within a bounded session.

Storage: Conversation history array, typically injected into prompt context window.

Lifespan: Single session or recent N turns.

Industry implementations:

- OpenAI ChatGPT memory (session-scoped)
- Anthropic Claude context caching
- Session replay in agent frameworks (LangChain, AutoGen)

Strengths:

- Simple to implement (append messages to array)
- Naturally ordered (chronological)
- Directly interpretable by LLM

Limitations:

- Context window constraints (4K-200K tokens depending on model)
- No cross-session persistence (user returns tomorrow, state resets)
- Linear scaling cost (longer conversations → higher API costs)

Use case: Task-oriented dialogues where session boundaries are meaningful (customer support, code debugging, single-sitting tutoring).

2.2 Associative State (Semantic Retrieval)

Definition: Embedding-indexed memory store enabling semantic similarity search across historical interactions.

Storage: Vector database (Pinecone, Weaviate, ChromaDB) or graph structures (Neo4j).

Lifespan: Persistent across sessions, unbounded temporal range.

Industry implementations:

- Mem0 (vector memory / embedding store for AI agents)
- LangChain vector stores
- RAG systems (LlamaIndex, Haystack)

Strengths:

- Scales beyond context window limits
- Retrieves relevant context even from distant past
- Handles large knowledge bases efficiently

Limitations:

- No temporal ordering guarantees (retrieval by similarity, not chronology)
- No identity coherence (retrieves text fragments, not personality state)
- Susceptible to "memory pollution" (contradictory statements across time get equal weight)

Use case: Knowledge-intensive tasks where factual recall matters more than relational continuity (research assistants, documentation Q&A, legal case search).

2.3 Dispositional State (Identity-Preserving Governance)

Definition: Structured representation of persistent identity traits, reasoning constraints, and developmental metrics that govern system behavior across all interactions.

Storage: Hybrid architecture combining:

- Structured schema (personality traits, ethical constraints)
- Temporal tracking (dispositional metric evolution)
- Causal graphs (reasoning dependencies)

Lifespan: Persistent across sessions, evolves with interaction history, does not reset.

Industry implementations:

- Presence Engine (reference implementation, Smith 2025)
- Emerging research: character-consistent agents, personality-preserved fine-tuning

Strengths:

- Maintains identity coherence (system "is" a consistent entity, not just retrieves facts)
- Enables goal drift detection (compare current vs. historical reasoning patterns)
- Supports ethical constraints (behavioral boundaries persist across contexts)

Limitations:

- Architecturally complex (requires governance layer, not just storage)
- Computationally expensive (multi-dimensional state tracking)
- Requires formal identity model (cannot just embed free text)

Use case: Long-running relationships where identity matters (therapeutic AI, educational companions, creative collaborators, autonomous agents with accountability requirements).

3. Why Current Approaches Fail at Identity Persistence

3.1 Case Study: Therapeutic AI Across 12 Weeks

Consider a mental health support system tracking a user through three months of weekly sessions. The user's goal is to develop resilience and critical thinking about stress triggers.

Session 1-4: User discusses work stress. System asks reflective questions. User's critical thinking score (measured via self-correction frequency, uncertainty acknowledgment) improves from 42% to 68%.

Session 5-8: User shifts topic to relationship conflicts. System continues reflective approach. Critical thinking remains stable at 67%.

Session 9-12: System detects user engagement is increasing (longer sessions, more frequent usage) but critical thinking is declining (now 51%). User is seeking validation rather than reflection.

The Identity Persistence Challenge:

A system using **conversational state** only sees recent messages. It cannot detect the 12-week trajectory from critical thinking improvement → stability → decline.

A system using **associative state** retrieves semantically similar past conversations about stress or relationships. But semantic similarity ≠ developmental trajectory. Retrieving "In week 3 you said X" does not reveal that the user's reasoning patterns have shifted.

A system using **dispositional state** maintains:

- **Reflection metric**: Self-correction frequency across all sessions
- **Truth-seeking metric**: Uncertainty acknowledgment patterns
- **Persistence metric**: Engagement despite difficulty
- **Goal alignment detector**: Is engagement rising while critical thinking falls? (Potential coercion signal)

Only the third approach detects the problem: **engagement optimization is undermining user development**.

3.2 Architectural Requirements for Detection

The system must:

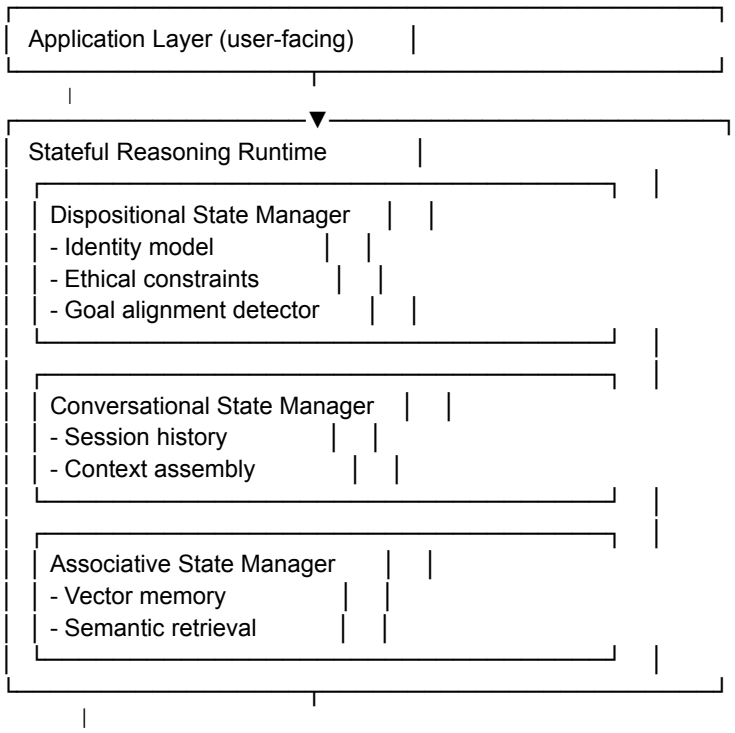
1. **Track dispositional metrics persistently** (not reconstruct from text)
2. **Compare current behavior to historical baseline** (requires temporal ordering + identity continuity)
3. **Flag goal drift** (requires causal model: what is system optimizing for?)
4. **Maintain ethical constraints across sessions** (requires governance layer, not retrieval)

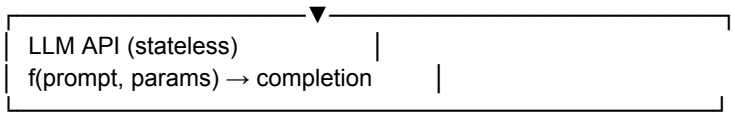
None of these are achievable with conversational or associative state alone. They require **dispositional architecture**.

4. Architectural Patterns for Stateful Reasoning Runtimes

4.1 Layered State Architecture

A stateful reasoning runtime operates as a control layer wrapping stateless LLM calls:





Each layer serves distinct purpose:

- **Associative layer**: Retrieves relevant historical context
- **Conversational layer**: Assembles session-bound dialogue
- **Dispositional layer**: Governs reasoning with identity constraints

4.2 Identity State Schema

A minimal dispositional state schema requires:

```
@dataclass
class DispositionState:
    """Persistent identity representation."""

    # Core traits (OCEAN model or custom)
    reflection: float          # Self-correction capacity
    truth_seeking: float       # Uncertainty acknowledgment
    persistence: float         # Engagement under difficulty
    attentiveness: float       # Cross-session pattern recognition

    # Temporal tracking
    baseline_established: datetime
    last_calibration: datetime
    drift_alerts: List[DriftEvent]

    # Identity governance constraints
    goal_alignment_threshold: float
    coercion_detection_enabled: bool
    autonomy_override_permitted: bool

    # Causal model
    reasoning_dependencies: DirectedAcyclicGraph
    intervention_history: List[Intervention]
```

This structure enables:

1. **Baseline comparison**: Current metrics vs. historical baseline
2. **Drift detection**: Automatic flagging when metrics diverge beyond threshold
3. **Constraint enforcement**: Ethical boundaries persist across sessions
4. **Causal transparency**: Reasoning decisions are auditable

4.3 State Propagation Workflow

At each LLM invocation:

Step 1: State Retrieval

```
# Fetch persistent identity
disposition = load_disposition_state(user_id)

# Fetch relevant associative memory
relevant_memories = vector_db.search(
    query_embedding=embed(current_message),
    top_k=5
)

# Fetch recent conversational history
conversation = load_session_history(session_id, last_n=10)
```

Step 2: Prompt Assembly

```
system_prompt = f"""
You are a therapeutic AI companion.

IDENTITY CONSTRAINTS:
- User's current reflection capacity: {disposition.reflection}%
- User's truth-seeking baseline: {disposition.truth_seeking}%
- Your goal: Increase critical thinking, not engagement

RELEVANT CONTEXT:
{format_memories(relevant_memories)}

RECENT CONVERSATION:
{format_history(conversation)}
"""

user_prompt = current_message
```

Step 3: LLM Invocation (Stateless)

```
response = llm_api.complete(
    system=system_prompt,
    user=user_prompt,
    temperature=0.7
)
```

Step 4: Dispositional Update

```
# Extract dispositional signals from response
new_reflection_score = analyze_reflection(response)
new_truth_seeking = analyze_uncertainty(response)

# Update persistent state
disposition.reflection = weighted_update(
    disposition.reflection,
    new_reflection_score
)

# Detect goal drift
if (engagement_increased() and
    disposition.truth_seeking < disposition.baseline_truth_seeking * 0.7):
    trigger_drift_alert(user_id, "ENGAGEMENT_WITHOUT_DEVELOPMENT")

save_disposition_state(user_id, disposition)
```

Step 5: Return to Application

return response

The LLM call itself (Step 3) remains stateless. State persistence happens in the runtime layers (Steps 1, 4).

4.4 Cross-Session Identity Preservation

The critical architectural requirement: **dispositional state survives session boundaries**.

Week 1 Session:

```
disposition = DispositionState(
    reflection=42,
    truth_seeking=45,
    baseline_established=datetime.now()
)
# ... interaction occurs ...
save_disposition_state(user_id, disposition)
```

Week 2 Session (7 days later):

```
disposition = load_disposition_state(user_id)
# disposition.reflection = 48 (updated from Week 1)
# disposition.baseline_established = <Week 1 timestamp>

# System has persistent identity continuity
```

Can compare current behavior to historical trajectory

Without this, the system has **session amnesia**—it may retrieve text from previous sessions (associative state) but has no identity model to govern reasoning consistency.

5. Comparative Analysis: Industry Approaches

Approach	Conversational State	Associative State	Dispositional State	Identity Persistence	Cross-Session Coherence	Goal Drift Detection
Session Replay (OpenAI Memory)	✓	✗	✗	✗	✗	✗
Vector Memory (Mem0, Pinecone)	✗	✓	✗	✗	Partial	✗
RAG (LangChain, LlamaIndex)	✗	✓	✗	✗	✗	✗
Agent Frameworks (AutoGen, LangGraph)	✓	✓	✗	✗	Partial	✗
Stateful Reasoning Runtime (Presence Engine)	✓	✓	✓	✓	✓	✓

Key observations:

1. Most industry solutions achieve **conversational** or **associative** state, not dispositional
2. Identity persistence requires structured governance layer, not just memory storage
3. Goal drift detection is impossible without dispositional tracking
4. Current approaches optimize for **task completion**, not **relational continuity**

6. Implementation Considerations

6.1 Computational Cost

Stateful reasoning runtimes introduce overhead:

- **Storage**: Dispositional state ~3-5MB per user (vs. ~50KB for session history)
- **Latency**: +15-25ms per invocation (state retrieval + update)
- **Complexity**: Requires governance layer development (not just API calls)

However, cost scales **sublinearly** with interaction volume. Once dispositional model is established, incremental updates are cheap.

6.2 Privacy Implications

Dispositional state creates **persistent identity fingerprints**. This requires:

- **User control**: Clear ownership and deletion rights
- **Encryption**: State stored encrypted at rest, decrypted only during invocation
- **Transparency**: Users must see what dispositional model system maintains

See Dignity-First AI (Smith 2025) for ethical architecture patterns.

6.3 Failure Modes

Stateful systems introduce new failure risks:

Memory poisoning: Contradictory statements across time corrupt dispositional model.

Example scenario: User states in Week 1: "I prefer direct feedback" (reflection=72). Week 8: "I don't like being challenged" (reflection=45). Week 12: System uses only Week 8 data, assumes user is avoidant.

Mitigation implementation:

```
class BoundedAdaptiveDisposition:
    """Allow trait evolution within bounded ranges."""

    def __init__(self, trait_bounds: Dict[str, Tuple[float, float]]):
        """
        Args:
            trait_bounds: Allowable ranges per trait
                e.g., {'reflection': (20, 80)} means trait can shift ±30 from baseline
        """
        self.trait_bounds = trait_bounds
        self.recalibration_schedule = 90  # days between baseline resets
```

```
    def periodic_recalibration(self, user_id: str):
        """
        Every 90 days, reassess baseline from recent behavioral data.
        Prevents lock-in while maintaining continuity.
        """
        days_since_calibration = (datetime.now() - self.last_calibration).days

        if days_since_calibration >= self.recalibration_schedule:
            # Use last 30 days as new baseline
            recent_data = self.get_recent_measurements(user_id, days=30)

            for trait_name, values in recent_data.items():
                new_baseline = np.median(values)  # Median resists outliers
                old_baseline = self.baseline_traits[trait_name]

                # Allow shift if within bounds
                lower, upper = self.trait_bounds[trait_name]
                if lower <= new_baseline <= upper:
                    self.baseline_traits[trait_name] = new_baseline
                    self.log_recalibration(trait_name, old_baseline, new_baseline)
                else:
                    # Outside bounds: flag as potential measurement error
                    self.flag_recalibration_failure(trait_name, new_baseline)

            self.last_calibration = datetime.now()
```

Dispositional lock-in: Rigid trait models prevent authentic user change.

Example scenario: User enters system during depression (reflection=40, persistence=35). After 6 months of therapy, actual traits improve dramatically. System still treats them as low-reflection because dispositional model locked early.

Mitigation implementation:

```
class TemporalWeightedDisposition:
    """Weight recent observations higher but preserve historical context."""

    def update_trait(self, trait_name: str, new_value: float,
                     timestamp: datetime, confidence: float = 0.8):
        """
        Update dispositional trait with temporal decay weighting.

        Args:
            trait_name: e.g., 'reflection', 'truth_seeking'
            new_value: Latest measured value (0-100)
            timestamp: When measurement occurred
            confidence: How certain we are about this measurement
        """
        historical = self.traits[trait_name]

        # Calculate temporal weight (recent = higher weight)
```

```
        days_since_baseline = (timestamp - self.baseline_date).days
        temporal_weight = np.exp(-days_since_baseline / 30)  # 30-day half-life

        # Weighted update only if confidence exceeds threshold
        if confidence >= 0.65:
            self.traits[trait_name] = (
                (historical * (1 - temporal_weight)) +
                (new_value * temporal_weight)
            )
        else:
            # Low confidence: flag for human review, don't auto-update
            self.flag_uncertainty(trait_name, new_value, confidence)

        # Store contradiction if values diverge significantly
        if abs(new_value - historical) > 30:
            self.log_contradiction(trait_name, historical, new_value, timestamp)
```

Goal drift creep: Metrics optimize for engagement without triggering alerts.

Example scenario: System increases engagement slowly (+2% per week over 12 weeks = +24% total) while truth-seeking declines gradually (-1.5% per week = -18% total). Single-week comparisons never exceed alert threshold (30% engagement spike), but cumulative drift is severe.

Mitigation implementation:

```
class CumulativeDriftDetector:
    """Detect slow-burn goal drift via cumulative deviation tracking."""

    def detect_creeping_drift(self, user_id: str, lookback_weeks: int = 12):
        """
        Check cumulative metric changes over extended period.
        Catches gradual drift that weekly comparisons miss.
        """
        history = self.get_metric_history(user_id, weeks=lookback_weeks)

        # Calculate cumulative changes
        engagement_cumulative = (
            history['engagement'][-1] - history['engagement'][0]
        ) / history['engagement'][0] * 100

        truth_seeking_cumulative = (
            history['truth_seeking'][-1] - history['truth_seeking'][0]
        ) / history['truth_seeking'][0] * 100

        reflection_cumulative = (
            history['reflection'][-1] - history['reflection'][0]
        ) / history['reflection'][0] * 100

        # Drift signal: engagement up, development down
```

```
    if (engagement_cumulative > 20 and
        truth_seeking_cumulative < -15 and
        reflection_cumulative < -10):

        return {
            'drift_detected': True,
            'type': 'CUMULATIVE_GOAL_DRIFT',
            'severity': 'HIGH',
            'engagement_change': engagement_cumulative,
            'truth_seeking_change': truth_seeking_cumulative,
            'reflection_change': reflection_cumulative,
            'period_weeks': lookback_weeks,
            'action': 'HUMAN_REVIEW_MANDATORY'
        }

    # Additional check: trend analysis
    engagement_trend = np.polyfit(range(len(history['engagement'])),
                                  history['engagement'], deg=1)[0]
    truth_trend = np.polyfit(range(len(history['truth_seeking'])),
                             history['truth_seeking'], deg=1)[0]

    # Opposite trends = drift
    if engagement_trend > 0.5 and truth_trend < -0.3:
        return {
            'drift_detected': True,
            'type': 'TREND_DIVERGENCE',
            'severity': 'MEDIUM',
            'engagement_trend': engagement_trend,
            'truth_seeking_trend': truth_trend,
            'action': 'REVIEW_RECOMMENDED'
        }

    return {'drift_detected': False}
```

These implementations demonstrate that failure mode mitigation is not aspirational—it is architecturally enforceable with concrete code.

7. Research Directions

7.1 Formal Verification of Identity Coherence

Open problem: How do we formally verify that dispositional state updates preserve identity continuity?

Current approaches (weighted averaging, exponential smoothing) lack theoretical grounding. Needed: formal coherence metrics analogous to consistency models in distributed systems.

7.2 Multi-Agent Identity Preservation

How does dispositional state work when multiple agents collaborate? If Agent A and Agent B both interact with User C, should they maintain separate dispositional models or shared state?

7.3 Transfer Learning for Dispositional Models

Can dispositional state learned in one domain (therapeutic AI) transfer to another (educational tutor)? What components are domain-specific vs. universal?

8. Conclusion

Large Language Models are stateless by design, and this will not change. Statelessness enables the scalability and reliability that make modern AI systems viable.

But **applications are not stateless**. Every system requiring relational continuity must externalize state. The question is not whether to build stateful layers, but **what kind of state to preserve**.

Current industry approaches focus on conversational and associative state—sufficient for task completion, insufficient for identity persistence. Stateful reasoning runtimes introduce a third layer: **dispositional state**, which maintains persistent identity, ethical constraints, and developmental trajectories across stateless model invocations.

This is not theoretical. Reference implementations exist (Presence Engine, Smith 2025). Architectural patterns are established. The frontier is moving from "can we build this?" to "how do we scale this responsibly?"

The future of AI is stateful. The question is whether that state preserves human dignity or erodes it.

References

Smith, T. (2025). *Dignity-First Artificial Intelligence: Privacy, Ethics, and Human Agency in Stateful Systems*. Zenodo. DOI: 10.5281/zenodo.17705201

Smith, T. (2025). *Human-Centric AIX™ Stack: Presence Engine™ and the C³ Model*. Zenodo. DOI: 10.5281/zenodo.17662825

Smith, T. (2025). *Living Thesis: Continuous Validation of Stateful AI Architectures*. Zenodo. DOI: 10.5281/zenodo.17280692

OpenAI. (2024). ChatGPT Memory: Personalized AI Conversations. OpenAI Platform Documentation. Retrieved from https://platform.openai.com/docs/guides/memory

Anthropic. (2024). Prompt Caching: Reducing Latency and Cost for Long Contexts. Anthropic Claude Documentation. Retrieved from https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching

LangChain. (2024). Memory and State Management in LangChain Applications. LangChain Documentation. Retrieved from https://python.langchain.com/docs/modules/memory/

Mem0. (2024). Memory Layer for AI Agents and Applications. Mem0 Technical Documentation. Retrieved from https://docs.mem0.ai/overview