Institutional Memory as Operational Infrastructure: A Framework for Context Persistence, Validity, and Decision Traceability in AI Operating Systems

November 2026

Abstract

An enterprise agent may retrieve an authentic document and still apply an instruction that is expired, superseded, or restricted to another customer. This paper examines how institutional knowledge becomes suitable for operational use in an AI operating system. It distinguishes remembered information from authorised instructions and proposes a lifecycle in which knowledge is captured, attributed, validated, approved, applied, and reconsidered. The analysis combines published work on language model memory and provenance with a constructed commercial scenario. Its contribution is a decision record that connects evidence, applicability, authority, and subsequent correction. It also develops a research protocol for testing whether these controls prevent invalid commitments without creating excessive review work. MatruOS provides an architectural context for the proposed requirements. The paper presents conceptual analysis and a proposed evaluation design; it reports no measured deployment results. Its central conclusion is that operational memory requires continuing stewardship, including a reliable account of what the business knew and permitted when an action occurred.

Keywords: institutional memory, provenance, enterprise AI, knowledge validity, policy versioning, decision traceability.

1. The problem of using remembered information

A distributor grants a customer a temporary discount to resolve a delayed delivery. Three months later, an agent retrieves the correspondence while preparing another quotation. The correspondence is genuine, the customer identity is correct, and the historical discount is accurately extracted. Applying that discount to the new order would nevertheless require authority that the original concession did not provide.

This example identifies a distinction between retrieval accuracy and operational validity. Retrieval establishes that information is relevant to a question. Operational validity concerns whether the information is applicable to the present action, whether its source has the necessary authority, and whether subsequent events have changed its meaning. An enterprise memory service needs to support both judgements.

Institutional memory includes policies, recorded decisions, customer commitments, explanations of exceptions, and knowledge of how work is performed. These materials differ in their status. A policy approved by finance, an employee’s description of usual practice, and an observation inferred by a model may all concern the same workflow. Their presence in a common search index does not make them equally authoritative.

The research question is therefore how an AI operating system should retain and retrieve business knowledge while preserving the conditions under which that knowledge can influence action. The unit of analysis is a knowledge item used in a consequential decision. The proposed framework applies to customer relationship management, enterprise resource planning, procurement, and related business workflows.

2. Research basis and analytical method

Packer and colleagues' MemGPT work investigates movement between memory tiers to extend usable context for language model applications. Its contribution concerns context management in document analysis and conversations. For enterprise execution, a further question is whether information brought into context remains authorised for the proposed action. [1]

Liu and colleagues show that the position of relevant information can affect performance on long context tasks in the models they evaluated. This supports testing retrieval and presentation arrangements rather than assuming that placing an entire history in a prompt ensures reliable use. The finding should be interpreted within the study's tasks and models. [2]

The W3C provenance model provides concepts for describing entities, activities, agents, and their relationships in the production of information. These concepts support an account of where a business record came from and who was involved. Provenance represents an account of origin; separate controls are needed to decide whether the recorded claim is correct and currently applicable. [3]

The present paper uses these sources as a focused conceptual foundation. It is not a systematic review of every memory architecture. The contribution is an operational synthesis: a lifecycle and evaluation design that connect provenance and retrieval to permission to act. The commercial examples are constructed to expose failure conditions. No customer observation or performance result is implied.

3. A knowledge record with operational meaning

3.1 Distinguishing types of knowledge

The framework separates descriptive records, interpretations, and executable policy. A descriptive record states what occurred, such as a customer receiving a concession on a particular order. An interpretation explains a possible reason or implication. Executable policy defines an authorised rule for a specified class of decisions.

An agent may assist in creating an interpretation, but promotion to executable policy should require a defined approval process. Repetition alone is insufficient. A recurring workaround may reveal a process defect rather than an accepted rule. A human reviewer should see the underlying examples and determine whether a policy change is justified.

Each item should identify its type so that retrieval can respect the distinction. A quotation agent could use a historical concession to ask whether an exception is warranted. It should not treat that history as permission to grant a concession automatically. The workflow can preserve useful context without extending its authority.

3.2 Recording identity and applicability

A knowledge item needs a stable identifier, an owner, its source, and the business entities to which it applies. Applicability may depend on customer, product, branch, order date, transaction value, or a specific agreement. These conditions should be represented explicitly wherever they can be expressed as structured rules.

Two time dimensions are relevant. One records when a policy was effective in the business. The other records when the system learned about it. A pricing instruction issued on Monday but entered on Wednesday creates different questions for a transaction processed on Tuesday. Retrospective investigation must be able to distinguish the policy's effective date from the system's awareness of it.

The framework also records approval status, expiry conditions, superseding records, and unresolved disputes. A document with an apparent signature may still require validation of the approver’s role. A record marked current may contain an unresolved contradiction. These conditions should be visible to the execution service as structured status, rather than buried in narrative text.

3.3 Keeping evidence accessible

A summary should link to the material from which it was derived. If an employee corrects an extracted amount or date, the correction should retain both the original extraction and the reason for revision where retention is appropriate. This allows a reviewer to understand the transformation rather than seeing only its final output.

The relevant evidence may include a document version, a conversation excerpt, or a transaction record. Access to that evidence must remain subject to permissions. A policy may be broadly available while the customer correspondence that prompted it is restricted. The system should support the authorised policy without exposing unrelated source material.

4. The lifecycle of operational memory

4.1 Capture and validation

Capture begins when a document, conversation, or decision enters the system. The service identifies its source and proposes structured fields. Validation checks entity identity, dates, quantities, and relationships. Where the input is ambiguous, the proposed item remains unresolved rather than becoming an active instruction.

For example, “use the agreed rate for this batch” does not establish which rate, batch, or agreement is intended. A useful clarification request identifies the missing fields. A general request to confirm the entire summary places unnecessary interpretation work on the reviewer and may allow an unnoticed omission to pass.

Validation should be proportional to consequence. A spelling correction in a customer note differs from a change to a payment condition. The process can allow routine editorial corrections while requiring explicit approval for fields that affect money, access, or obligations. This distinction should be enforced by the application.

4.2 Approval and publication

Approval establishes who authorises the item and for what purpose. Publication makes the approved version available to eligible workflows. A proposed rule should not appear as active merely because it has been indexed for search. Search infrastructure and policy activation therefore need separate state transitions.

The approving person should review the structured conditions and examples of their application. If a rule concerns discounts above a threshold, the review should show what happens at the threshold and when several discounts apply to one order. Examples help reveal differences between the wording of a policy and its operational interpretation.

Publication should create a version that can be referenced by later decisions. Editing an active record in place can obscure which instruction an agent used. A revision should preserve the relationship between versions and establish how pending work is affected.

4.3 Retrieval and use

Before retrieval, the service should determine the business entity and the user's permitted scope. It should then identify eligible knowledge items using structured applicability conditions. Semantic retrieval can help find relevant supporting material within that scope. This order reduces dependence on the model to discover access restrictions after receiving the information.

At the time of action, the execution service should confirm that the selected instruction remains applicable. A policy can change between retrieval and commitment. A cached result therefore needs a version and a method for detecting relevant change. A recheck should consider the fields on which the decision depends, such as customer status or order value.

The record of use connects the selected policy, supporting evidence, approval, and resulting action. This allows later investigation of both successful and disputed decisions. It also makes it possible to identify transactions that may require review after a knowledge correction.

4.4 Correction, expiry, and withdrawal

A correction can affect future actions and earlier commitments differently. If a policy was incorrectly entered, stopping its future use is an immediate containment step. Existing orders require a separate assessment because the business may already have communicated terms or fulfilled part of the obligation.

The framework therefore proposes an impact query: which decisions depended on the corrected item, which remain open, and which have external consequences? The answer supports targeted review. Automatic reversal would be inappropriate where the business needs to honour a commitment or communicate a change.

Expiry should also distinguish historical retention from operational availability. An expired instruction may remain necessary for explaining a previous decision. It should be excluded from current authority while remaining retrievable by authorised reviewers. Deletion and retention requirements need separate organisational assessment.

5. A worked commercial scenario

Assume that a customer normally receives a three percent discount. A five percent concession is authorised for one delayed order, valid until the end of a stated month. A later email asks the sales team to "continue on the same terms," but the sender is a customer contact without authority to change the supplier's pricing policy.

The memory service records the standard policy, the exceptional approval, and the later request as different items. The exceptional approval is linked to the original order and its validity period. The later request is evidence of customer intent and can initiate a review. It cannot by itself amend the authorised policy.

When the quotation agent prepares a new order, it retrieves the standard terms and the historical exception as context. The execution service checks whether an active approval applies to the new order. If none exists, the agent prepares a quotation using the standard policy or requests approval for the proposed exception, depending on the user's instruction.

Suppose the manager approves an exception for the new order but changes the maximum eligible quantity. The approval must bind to that quantity and order version. If the quantity subsequently changes, the earlier approval should be reconsidered according to policy. A natural language statement that the manager "already approved it" is insufficient evidence for the changed request.

This scenario yields a testable proposition: explicitly separating historical evidence, customer requests, and executable authority should reduce invalid reuse of concessions. It also introduces a cost to measure, since additional classification and review may increase preparation effort. The research design should evaluate both effects.

6. Architecture and application to MatruOS

A practical implementation can store structured knowledge metadata beside the operational records it references, while keeping source documents in controlled storage. Search indexes support discovery but should not become the sole authority for current status. An index update delay must not leave a withdrawn rule executable.

MatruOS provides an application context in which pricing, customer records, and order activity can be connected to policy records. The proposed design would make policy versions visible to quotation and approval workflows. It would also record when an employee overrides a suggested interpretation and why. These are requirements for evaluation, not claims of independently verified product behaviour.

The system needs ownership of maintenance. Finance may own credit policy, sales may maintain commercial agreements, and operations may own delivery constraints. A shared memory service should preserve those responsibilities. A general administrator should not acquire substantive policy authority simply by having access to the configuration screen.

Model replacement adds another maintenance question. Summaries, extracted fields, and retrieval rankings may change when a model changes. Existing approved policy should remain stable until deliberately revised. A new model should be evaluated against historical cases with known applicability decisions, including cases where abstention or escalation was required.

7. Proposed evaluation protocol

The evaluation set should contain ordinary policies and deliberately difficult cases: expired concessions, conflicting instructions, misidentified customers, incomplete dates, delayed ingestion, and later corrections. Each case needs a documented expected decision and supporting evidence agreed by qualified business reviewers.

The principal measure is the proportion of actions that use a currently valid and authorised instruction. Additional measures include invalid acceptance, unnecessary rejection, unresolved conflict detection, and completeness of decision traceability. Retrieval relevance should be measured separately so that an accurate search result does not conceal an invalid action.

A controlled test can compare a retrieval service using documents alone with the same service using structured applicability and approval records. Both conditions should use the same underlying model, task set, and allowed tools. The purpose is to isolate the effect of the proposed controls rather than attribute a change caused by different models to memory design.

Temporal cases should be tested explicitly. Introduce a policy change after retrieval but before execution, then observe whether the action is revalidated. Introduce a correction after commitment and inspect whether affected records are discoverable. Repeat tests with changed document order and irrelevant material to assess sensitivity to context presentation.

Review effort belongs in the results. Record time spent resolving ambiguity, maintaining policy metadata, and investigating a disputed action. A memory design that prevents invalid reuse by referring every request to a manager may be safe within the test but operationally unsuitable. The evaluation should report where human work is concentrated and whether the workload remains manageable.

8. Limitations and research implications

The framework assumes that the organisation can identify responsible policy owners. Where authority is disputed or undocumented, the software cannot resolve that organisational question by inference. Implementation may first require agreement about who can authorise particular commitments.

Structured conditions also have limits. Some commercial judgements depend on relationships or circumstances that are difficult to encode. The system should expose that limitation and route the decision to an authorised person. It should preserve the explanation without automatically generalising the decision into a rule.

The proposed evaluation uses constructed cases and would need extension to operational data before claims about deployment effects could be made. Historical cases may omit the conversations that originally influenced a decision. Reviewer disagreement should therefore be recorded as uncertainty in the reference outcome, not silently resolved in favour of the system.

The contribution is a testable connection between memory lifecycle controls and the validity of business action. It does not establish a new general theory of memory or claim that the underlying provenance concepts are novel. Subsequent work should examine whether maintenance effort, retrieval quality, and action validity remain acceptable as the knowledge base changes.

9. Conclusion

Institutional memory becomes operational infrastructure when it can support a defensible decision about what information applies, who authorised its use, and what changed afterwards. Storage and retrieval provide necessary capabilities, while continuing ownership preserves meaning and authority.

The proposed lifecycle connects capture, validation, approval, execution, and correction. Its value should be tested through invalid commitments prevented, valid work completed, and the human effort required to maintain the record. An AI operating system should be able to explain the knowledge available at the time of action and identify earlier decisions affected by a later correction.

For MatruOS and other enterprise implementations, this provides a concrete research agenda: evaluate memory through the business decisions it supports, including the decisions it must defer. Reliable institutional knowledge depends on maintained relationships between evidence, policy, and responsibility.

Author affiliation and disclosure

The author is affiliated with Code Colonies Private Limited, Ahmedabad, and leads the MatruOS team. MatruOS is an architectural application context. No independent product evaluation is reported.

References

[1] Packer, C., et al. (2024). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560, version 2.

https://arxiv.org/abs/2310.08560v2

[2] Liu, N. F., et al. (2023). Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172, version 3.

https://arxiv.org/abs/2307.03172v3

[3] Moreau, L., and Missier, P., editors (2013). PROV-DM: The PROV Data Model. W3C Recommendation, 30 April.

https://www.w3.org/TR/prov-dm/