Continuity-Bearing Memory in Practice

An Addendum to "Beyond Chat Memory: Layered Memory Architectures for Persistent Expert Systems"

Jordi Buskermolen

May 2026

Note on This Addendum

This document is a companion to "Beyond Chat Memory: Layered Memory Architectures for Persistent Expert Systems" (Buskermolen, April 2026), hereafter referred to as the original paper. It is not a revision. The original paper's category claim, its identification of the continuity burden, the requirements R1–R7, the failure modes F1–F6, the layered reference architecture, the promotion pathways P1–P5, and the evaluation dimensions E1–E6 stand as published.

The purpose of this addendum is narrower. Since the original paper was written, the public availability of a maturing class of open-source agent memory systems—typified by Garry Tan's GBrain, an opinionated personal agent memory toolkit published in early 2026—has made it possible to anchor the original paper's architectural argument against working implementations. These systems are not persistent expert systems in the sense defined by the original paper. They are general-purpose agent memory toolkits aimed at a different design target. But their engineering, particularly around relational extraction, retrieval evaluation, and durable operational substrate, encodes patterns that the layered architecture should absorb rather than reinvent.

This addendum makes four contributions. First, it proposes a design-target taxonomy that situates persistent expert systems alongside the production-agent class and clarifies why systems with similar primitives behave differently. Second, it identifies five integration patterns from the production-agent class that strengthen the layered architecture against the original paper's own requirements. Third, it identifies three patterns that should not be imported, and explains why each would compromise the continuity burden. Fourth, it proposes a dual-frame evaluation protocol that combines retrieval-quality benchmarks with the continuity-oriented dimensions E1–E6.

After this preface, the production-agent class is referred to abstractly rather than by product name. The patterns described are properties of a class of system, not endorsements of any particular implementation.

Abstract

The original paper argued that persistent expert systems impose a distinctive continuity burden—preserving identity, scope, current truth, and validated outcomes across time—and that recall-oriented

memory designs do not satisfy this burden without modification. It proposed a layered reference architecture and an evaluation frame oriented to continuity rather than retrieval accuracy alone.

This addendum extends that argument in light of recent production agent memory systems. It argues that the layered architecture can absorb five engineering patterns from the production-agent class—typed relational extraction, compiled-truth-and-timeline record structure, a durable operational substrate, tiered candidate triage, and dual-frame evaluation—without compromising its theoretical center. It further argues that three other patterns from the same class—open-ended ingestion, automated promotion by repetition, and agent self-modification of identity—cannot be imported without violating the continuity burden the original architecture was designed to carry. A revised view of the architecture is sketched in which the relational layer is cross-cutting rather than additive, and a dual-frame evaluation protocol is proposed in which retrieval-quality metrics calibrate developmental velocity while continuity-oriented metrics calibrate trustworthiness.

The broader claim is that the boundary between governed and auto-accumulating memory is not a question of better or worse design; it is a question of design target. Different domains warrant different settings of that dial, and the taxonomy proposed here is intended as a vocabulary for that conversation.

A. Framing: Why an Addendum Is Useful

The original paper is explicit about its register. It is a theory and position paper with an architectural proposal, not an empirical benchmark paper. Its central contribution is conceptual: defining the persistent-expert system category, identifying the continuity burden, and deriving an architectural response. It does not present implementations, performance numbers, or empirical comparisons, and the limitations section acknowledges this directly.

Since the paper was completed, the landscape of open-source agent memory systems has continued to mature. A particular class of system has emerged that deserves engagement: production agent memory toolkits intended to be installed into an existing agent runtime, exposing hybrid retrieval (vector and keyword fusion), automatic relational extraction over a typed knowledge graph, durable background work queues, and reproducible retrieval benchmarks. These systems are produced under different design constraints than the persistent expert systems the original paper addresses. They tend to be generalist rather than bounded, optimised for autonomous accumulation rather than governed promotion, and evaluated on retrieval-quality metrics rather than continuity-oriented ones.

This difference is not incidental. The original paper argues that memory designs reflect the design target they were built for, and that recall-oriented designs do not satisfy the continuity burden simply because they were never asked to. The same point applies in reverse: the production-agent class encodes engineering disciplines that the layered architecture would benefit from, but those disciplines were developed in service of a different target. The integration question is therefore not whether to import

features, but which features survive translation into a governed, bounded, continuity-bearing architecture, and which do not.

Three reasons motivate engaging with this question explicitly rather than implicitly.

First, empirical anchoring. The original paper's most foreseeable critique is that its argument is unfalsified rather than wrong: an architectural proposal without an instantiation cannot be evaluated. Engagement with the production-agent class supplies the missing reference points. The retrieval-quality benchmarks that this class has begun to publish provide a concrete calibration against which the architecture can be measured along one of its evaluation axes. The continuity-oriented dimensions E1–E6 remain the central evaluation frame, but they are no longer the only frame.

Second, contrast as clarification. A risk of the original paper's synthesis framing is that readers may assimilate its claims to whatever recall-oriented memory design they already know. The contribution becomes harder to see when nothing concrete pushes against it. The production-agent class supplies that contrast. The architectural moves that distinguish persistent expert systems—identity as a first-class memory object, governed promotion, validity states on temporal facts, scope-aware retrieval ordering—are most visible when set against a serious system that does not make those moves and is nevertheless effective at its own target. Contrast clarifies the original argument more sharply than additional internal exposition could.

Third, generativity. An architectural proposal is strengthened, not weakened, by demonstrating that it can absorb engineering from neighbouring systems. The integration patterns described below do not dilute the original architecture; they fit cleanly into its existing layers and governance pathways. This is itself evidence that the architecture was specified at the right level of abstraction. A reference architecture that could not absorb relational extraction or durable operational substrate would be a brittle specification. The fact that the layered architecture can absorb these patterns while keeping its theoretical center intact is a property worth noting.

The remainder of this addendum is organised as follows. Section B proposes a design-target taxonomy for agent memory systems. Section C describes five integration patterns and their fit with the layered architecture. Section D describes three patterns that should not be imported. Section E sketches a refined view of the architecture in which the relational layer is treated as cross-cutting. Section F proposes a dual-frame evaluation protocol. Section G discusses open questions and future work.

B. A Design-Target Taxonomy of Agent Memory Systems

The original paper situated persistent expert systems against adjacent categories—general assistants, retrieval-augmented systems, profile-memory products, transcript-replay systems—but did so primarily by reference to memory mechanism. A more productive partition runs along design target rather than

mechanism. Two axes appear sufficient to organise the current landscape: domain boundedness, and promotion discipline.

B.1 The Two Axes

Boundedness describes whether the system inhabits a defined domain of competence or operates across arbitrary subject matter. A bounded system declines out-of-scope queries; an unbounded system attempts to answer them. Boundedness is a property of the system's design, not its capability. An unbounded system may be excellent at many domains; what makes it unbounded is the absence of an enforced perimeter.

Promotion discipline describes how content enters durable memory. A high-discipline system promotes content into long-term memory through explicit pathways with human authority on the critical path. A low-discipline system accumulates durable memory automatically, often by frequency, recency, or co-occurrence signals. Promotion discipline is a separate property from boundedness: a bounded system can be high or low on discipline, and so can an unbounded system.

Together these axes produce four quadrants:

	Low promotion discipline	**High promotion discipline**
Unbounded domain	Generalist auto-accumulating agent memory (the production-agent class)	General assistant with workflow gating (rare; the typical position of enterprise assistants with light governance)
Bounded domain	Domain-specific agents with weak governance (a common failure mode; often appears as a chatbot that confidently invents content within its declared domain)	Persistent expert systems as defined by the original paper

This taxonomy does three things. First, it makes the original paper's category claim sharper. Persistent expert systems are not just "assistants with better memory"; they occupy the bounded–governed quadrant deliberately, and the architecture follows from that position. Second, it situates the production-agent class accurately. Those systems are not deficient persistent experts; they are excellent unbounded –auto-accumulating agents, and judging them by E1–E6 would be category-confused in the same way that judging a persistent expert by retrieval throughput alone would be. Third, it surfaces the bounded–low-discipline quadrant, which is structurally underspecified and tends to produce systems that look like persistent experts but drift unpredictably because their durable memory accumulates without governance.

B.2 What the Axes Imply for Memory Design

Each quadrant exerts different pressures on memory architecture, and the architectural moves that succeed in one quadrant do not necessarily transfer to another.

In the unbounded–low-discipline quadrant, memory architecture is dominated by the problem of coverage. The system must ingest broadly, link liberally, and retrieve across heterogeneous content. Relational extraction over a typed graph is highly valuable here because the relationships between entities provide a navigation surface that bounded vocabulary cannot. Auto-accumulation is acceptable because the cost of remembering a wrong fact is low: it is one entry among many, and downstream judgment-based components can filter at retrieval time.

In the bounded–high-discipline quadrant, memory architecture is dominated by the problem of trust over time. The system must remain the same expert, use current truths, and respect accepted outcomes across years of cumulative work. Auto-accumulation is unsafe because the cost of remembering a wrong fact is high: the fact will be retrieved as if it were authoritative, and the system's users will rely on it. Governed promotion is therefore not friction; it is the mechanism by which the system earns and keeps its authority.

This asymmetry is the operative observation. Engineering patterns that emerged to solve the coverage problem in the unbounded–low-discipline quadrant can often be imported into the bounded–high-discipline quadrant—provided that promotion discipline is preserved as the import happens. Engineering patterns that are themselves expressions of low promotion discipline (autonomous self-modification, mention-count promotion, open-ended ingestion) cannot be imported without changing the design target itself.

The five integration patterns described in Section C all satisfy the first condition: they are mechanism-level disciplines from the unbounded–low-discipline quadrant that survive translation into the bounded–high-discipline quadrant. The three patterns described in Section D fail it.

C. Five Integration Patterns

Each of the five patterns below is described in three parts: what the pattern does, which original-paper requirements (R1–R7) or failure modes (F1–F6) it strengthens, and what the limits of integration are. Implementation detail is omitted intentionally; this addendum stays at the architectural level the original paper occupied.

The patterns are referred to throughout as I1–I5 (integration patterns).

C.1 I1: Typed Relational Extraction as a Cross-Cutting Memory Layer

The pattern. On every write to durable memory, a deterministic extraction pass identifies entity references within the content and creates typed relationships between the writing record and the referenced entities. The relationship types are domain-specific and finite—for a coaching domain, types might include "applies-to-cohort", "evidenced-by-session", "produced-outcome", "supersedes", "authored-by". The extraction is reconciled on edit: when a reference is removed from a record, the corresponding relationship is removed; when a new reference is added, a new relationship is created. The

resulting structure is a typed graph layered over the existing memory store rather than replacing it. Retrieval is then permitted to consult the graph either as the primary access path (for relational queries) or as a ranking signal (well-connected entities are weighted higher).

Why it fits. The original paper proposes a layered architecture organised by the role each layer plays in continuity. The relational layer does not fit cleanly inside any one of the five layers, because relationships exist between records belonging to different layers: an outcome (Layer 3) is evidenced by sessions (Layer 5), applies to a scope (Layer 2), depends on a current fact (Layer 4), and may be authorised by an identity-core principle (Layer 1). Treating the typed graph as an additional layer would understate its role. It is more accurately understood as cross-cutting: a structure that connects records across the existing layers and makes the relationships between them explicit, queryable, and statused.

Which requirements it strengthens. I1 most directly strengthens R2 (scope-aware recall) and R5 (provenance and governability). Scope itself becomes a relationship type rather than a metadata field, which means scope membership can be queried, audited, and revised through the same mechanisms as any other relationship. Provenance, which in the original paper is described as JSON evidence attached to outcome records, becomes a first-class structure: the question “why does the system believe this?” is answered by traversing the typed edges that connect the belief to its sources. I1 also indirectly strengthens R6 (contradiction handling and supersession) because supersession is naturally represented as an edge type, and the supersession history of any record becomes a path through the graph rather than a chain of foreign keys.

Which failure modes it addresses. F1 (flat semantic retrieval vs. scoped relevance) is directly addressed: for relational queries, the graph provides a precision mechanism that vector search cannot match. F6 (opaque state vs. provenance and contradiction handling) is partially addressed by making provenance traversable rather than only attached. The pattern does not, on its own, address F2 (transcript replay vs. governed memory) or F3 (the original paper’s identity drift failure mode), which require governance mechanisms rather than relational structure.

Limits of integration. Two limits should be acknowledged. First, the value of the relational layer depends on the quality of the typed relationship vocabulary, which is itself a domain question. A poorly chosen vocabulary creates either too few types (relationships collapse into a single edge type and lose precision) or too many types (extraction becomes unreliable and the graph becomes inconsistent across records). The vocabulary must be designed deliberately for the domain, and must be governed in the same way the identity core is. Second, deterministic extraction works well for explicit references but poorly for implicit ones. A record that says “the pricing intervention” without naming a specific cohort cannot be linked to the cohort by regex; deterministic extraction must accept that some relationships will remain implicit. The architecture should treat the relational layer as a high-precision overlay, not as an exhaustive index.

C.2 I2: Compiled Truth and Timeline as a Record-Level Convention

The pattern. Every durable memory record is structured as two distinct sections rather than a single content field. The first section, the compiled truth, contains the system's current best understanding of whatever the record concerns. It is rewritable and is the section that ranks first in retrieval. The second section, the timeline, contains append-only dated entries documenting the observations, sources, and revisions that contributed to the compiled truth. The compiled-truth section receives a retrieval boost over the timeline section, so that the system's current understanding ranks above the historical evidence that produced it. When the compiled truth is revised, the previous version is moved into the timeline as a dated entry rather than discarded.

Why it fits. The original paper draws several distinctions that this pattern expresses at the record level: between history and memory (Section 3.2), between current and historical truth (Section 6.3), and between exploration and outcome (throughout). The compiled-truth-and-timeline convention encodes all three distinctions in the structure of a single record. It is the cleanest available expression of the original paper's claim that durable memory should be governed at the level of items, not only at the level of stores.

Which requirements it strengthens. I2 most directly strengthens R3 (temporal truth management), R4 (outcome continuity), and R7 (compression without collapse). R3 is strengthened because each record makes its own temporal status visible: the current section is current by virtue of its position, and the historical material is preserved without being confusable with current truth. R4 is strengthened because outcomes acquire a stable surface—the compiled truth—that downstream work can build on, while the evidence base remains inspectable below. R7 is strengthened because compression becomes structural rather than algorithmic: the compiled truth is the compressed representation, and the timeline is the uncompressed evidence, and there is no point at which the two are confused.

Which failure modes it addresses. F4 (stale truth reuse) is directly addressed: a system that retrieves the compiled truth and only consults the timeline when asked cannot accidentally surface a historical observation as if it were current consensus. F5 (outcome amnesia) is partially addressed: outcomes become more visible because they have a structurally privileged position in their own records. F6 (opaque state) is partially addressed because every record carries its own evidence trail in inspectable form.

Limits of integration. The convention requires that compiled truth and timeline genuinely separate. If the timeline accumulates content that should have been promoted into compiled truth, the system will quietly under-retrieve current truth. If the compiled truth is edited without a corresponding timeline entry recording the revision, the system loses the very property that distinguishes it from a flat note. Discipline in maintaining the split is itself a governance question, and the pattern works only when paired with explicit revision rules. The integration is therefore not just a schema change; it is a discipline that must be enforced at write time.

C.3 I3: Durable Operational Substrate for Memory Governance

The pattern. All long-running memory-governance work—pattern surfacing, supersession checks, candidate-outcome detection, archive compression, relational extraction backfills, statistical threshold evaluations—is dispatched to a durable job system rather than executed in request threads or ad-hoc cron processes. Jobs survive restarts, support retry and cancellation, expose parent–child relationships, and emit audit records on every state transition. A routing principle separates deterministic work (handled by the job system, with no model calls) from judgment work (handled by model-based components). Operational observability of the memory system—which jobs ran, which records were created or superseded, which candidates were generated—becomes a property of the substrate rather than a separately instrumented concern.

Why it fits. The original paper specifies governance pathways P1–P5 as the mechanism by which content enters durable memory. It is largely silent on how those pathways are operationalised. In practice, several of the pathways—P3 (repetition stability), P5 (outcome confirmation), and the candidate generation that precedes P2 (workflow approval)—are deterministic computations over the existing memory store. They are exactly the kind of work that requires durable execution, retry semantics, and audit records. Without a substrate, these pathways either fail silently under load, accumulate inconsistent state across partial runs, or sit in request threads and become unobservable. With a substrate, each governance action is a first-class operational event with its own lifecycle.

Which requirements it strengthens. I3 strengthens R5 (provenance and governability) at the operational layer. The original paper specifies that promotion pathways should be attributable and inspectable. A durable substrate makes them so: every promotion event has a job id, a timestamp, an attempt history, and a result that can be reviewed and audited. I3 also strengthens R6 (contradiction handling and supersession) because supersession is typically discovered by background analysis (a new fact is observed that contradicts an old one), and that discovery is exactly the kind of deterministic work the substrate handles well.

Which failure modes it addresses. None of the original paper's failure modes are operational. I3 does not address F1–F6 directly. What it addresses is a meta-failure not enumerated in the original paper: the failure of governance pathways to operate reliably at scale. A system whose architecture satisfies R1–R7 on paper but whose governance work is implemented as fragile cron jobs will exhibit the original paper's failure modes anyway, because the governance never reliably runs. I3 makes governance a property of the running system, not a property of the specification.

Limits of integration. The substrate is necessary but not sufficient. A durable substrate that dispatches the wrong jobs reliably is still wrong. The discipline that I3 enables is observability; the design of which governance work to run remains a separate question, addressed by the promotion pathways and the candidate-generation logic. The integration should be understood as moving governance from an architectural concept to an operational concern, not as replacing governance with operations.

C.4 I4: Tiered Triage for Promotion at Scale

The pattern. Candidate observations—the inputs to promotion pathways—are not all surfaced for review at the same urgency. A tiered system classifies candidates by the strength of the underlying signal: a single weak observation is held at a low tier and surfaced only on demand; a candidate that has recurred across several independent sources is held at a middle tier and surfaced in a regular review queue; a candidate that meets a higher threshold (across-source recurrence, statistical significance, or domain-specific criteria) is escalated for individual review. The tiering governs visibility and review priority, not authority. No tier auto-promotes; promotion still requires a human pathway. The mechanism is a triage layer, not a promotion layer.

Why it fits. The original paper specifies promotion pathways but does not address candidate volume. At small scale this is acceptable: a coach or operator can review all candidates. At larger scale—many users, many sessions, accumulating over years—the candidate stream exceeds human review capacity. The original paper's commitment to governed promotion remains correct, but governance becomes a bottleneck unless candidates can be prioritised. Tiered triage is the mechanism by which review attention is allocated rationally without surrendering the principle that humans hold authority over durable truth.

Which requirements it strengthens. I4 strengthens R5 (provenance and governability) at scale by making governance feasible in practice rather than only in principle. It also has a subtler relationship to R7 (compression without collapse): tiered candidates that are never promoted still exist in the system at their tier level, where they can be aggregated, summarised, and surfaced as weak signals when explicitly queried. The architecture thereby preserves the distinction between "not promoted" and "never observed," which is a kind of compression the original paper implicitly requires but does not specify.

Which failure modes it addresses. I4 does not address F1–F6 directly. It addresses a meta-failure adjacent to I3's: the failure of governance pathways to be exercised at all because the volume overwhelms reviewers. A system that has the right pathways but no triage will tend toward two equally bad outcomes: either reviewers stop engaging, and candidates accumulate unreviewed (silently expanding low-trust memory), or reviewers process candidates superficially, and promotion becomes rubber-stamping (silently expanding high-trust memory under low scrutiny). Tiered triage prevents both by aligning review attention with signal strength.

Limits of integration. The pattern requires that tier thresholds be set deliberately and reviewed over time. Thresholds that are too lax push too many candidates upward and recreate the volume problem; thresholds that are too strict suppress legitimate candidates and create the appearance of agreement where there is only inattention. Threshold maintenance is itself a governance question and should follow explicit pathways (analogous to P1 or P2), not silent inference from observed reviewer behaviour.

C.5 I5: Dual-Frame Evaluation

The pattern. Evaluation is conducted in two complementary frames rather than one. The first frame consists of retrieval-quality metrics over a fixed corpus with hand-labelled relevance judgments: precision at k, recall at k, mean reciprocal rank, and normalised discounted cumulative gain. These metrics are deterministic, fast, and reproducible. They run on every significant change to the retrieval stack and surface regressions immediately. The second frame consists of continuity-oriented metrics over multi-session protocols, evaluating the dimensions E1–E6 specified in the original paper. These metrics are slower, more expensive, and more interpretive, but they evaluate properties that retrieval metrics cannot reach. The two frames are reported together; neither replaces the other.

Why it fits. The original paper argues that recall-only benchmarks are incomplete for persistent expert systems. It does not argue that they are useless. Section 8 explicitly states that traditional recall benchmarks remain valuable and that the proposed continuity dimensions are complementary to them. The dual-frame protocol makes that complementarity operational. It also addresses a practical concern that the original paper acknowledges but does not resolve: continuity evaluations are inherently slow because they require multi-session protocols, which makes them poorly suited as a development feedback signal. Retrieval-quality metrics provide that feedback signal; continuity metrics provide the trustworthiness audit.

Which requirements it strengthens. I5 does not strengthen any individual requirement; it strengthens the architecture's evaluability as a whole. The original paper's evaluation frame is well-specified at the dimension level (E1–E6) but underspecified at the protocol level: the task sketches are illustrative rather than directly executable. The dual-frame protocol does not resolve that underspecification, but it reduces its consequence. A system whose retrieval-quality metrics regress can be investigated immediately; a system whose continuity metrics regress can be investigated quarterly. The combined signal is stronger than either alone.

Which failure modes it addresses. The pattern does not directly address F1–F6. It increases the probability that those failure modes are detected when they occur. F4 (stale truth reuse) and F6 (opaque state) in particular benefit from the addition of retrieval-quality metrics, because they often manifest as silent retrieval errors that continuity protocols catch only later.

Limits of integration. The dual-frame protocol requires a domain-specific evaluation corpus with hand-labelled relevance judgments. For persistent expert systems this is a non-trivial undertaking, because the corpus must reflect the system's actual scope rather than an off-the-shelf benchmark. Constructing the corpus is itself a knowledge-management exercise: the queries must span the domain, the relevance judgments must be made by domain experts, and the corpus must be revised as the underlying memory evolves. The cost is real, but the resulting feedback signal substantially shortens development cycles and creates an audit trail of retrieval-quality changes over time. It should be regarded as part of the system's infrastructure, not as an optional addition.

D. Patterns That Should Not Be Imported

Three patterns from the production-agent class do not survive translation into a bounded, governed architecture. Each is described briefly, with the architectural reason it fails to translate.

D.1 Open-Ended Ingestion

Production agent memory systems frequently ship integrations that ingest content from arbitrary sources—email, messaging platforms, social feeds, voice calls—into durable memory automatically. This is appropriate for an unbounded agent, where the value of memory is roughly proportional to coverage. It is inappropriate for a persistent expert system, where the architecture's value depends on scope discipline. Unrestricted ingestion violates condition C1 (specialised role) of the original paper's category definition. Even with downstream filtering, the cost of out-of-scope content in durable memory is asymmetric: a single off-topic record retrieved as if it were authoritative can produce a continuity failure that recall metrics will not catch. Ingestion into a persistent expert system should remain explicit, scoped, and governed.

D.2 Automated Promotion by Repetition

Some production systems treat repeated mention of an entity, fact, or pattern as a promotion signal: a piece of content that has appeared often enough is elevated to higher-tier memory automatically. This is one face of the auto-accumulation principle that defines the production-agent class. The original paper accepts repetition as a candidate signal (P3, repetition stability) but explicitly requires human confirmation before promotion. The distinction is load-bearing. Repetition can be produced by stable truth, by widespread agreement, or by a single noisy observation that happens to be referenced frequently in subsequent records. Without a human pathway, the architecture cannot reliably distinguish these cases, and silently expanding durable memory under any of them produces the failure mode the original paper calls outcome amnesia's mirror: outcome inflation, where the system treats undecided material as decided. Tiered triage (I4) supplies the legitimate use of repetition signal within a governed architecture; automated promotion does not.

D.3 Agent Self-Modification of Identity

The most operationally elegant pattern in the production-agent class is the discipline of converting recurrent failures into durable system improvements: when the agent fails in a particular way, the failure becomes a new skill or rule that prevents future occurrences. This is a powerful property in an unbounded system, where the system's scope is not itself a managed asset. It is incompatible with the persistent expert architecture, where the identity core (Layer 1) is governed and only modifiable through explicit promotion under SUPER_ADMIN-equivalent authority. The original paper is explicit on this: the identity core is a memory object, but its authority structure is more restrictive than other memory layers, and the system does not self-modify. Importing agent self-modification of identity is not an enhancement; it is a

redefinition of the design target. A persistent expert system that can rewrite its own identity is no longer a persistent expert system in the sense the original paper defined.

E. A Refined View of the Architecture

The integration patterns in Section C do not require a structural revision of the layered architecture, but they do clarify one aspect of it. The original paper presents the architecture as five sequential layers (identity core, scoped working memory, episodic and outcome memory, temporal fact memory, deep archive) with retrieval moving from highest authority to lowest. This presentation is correct for the question of retrieval order, which is what the original paper foregrounds. It understates two other dimensions that become relevant once integration is considered.

E.1 The Relational Layer Is Cross-Cutting

The typed relational layer introduced in I1 does not occupy a position in the retrieval-order stack. It connects records belonging to different layers and is consulted in addition to layered retrieval rather than instead of it. A refined diagram would draw the five layers as a vertical stack ordered by retrieval authority, and the typed graph as a horizontal structure spanning them. The graph contains edges whose endpoints belong to different layers: an outcome (Layer 3) is connected to a session (Layer 5), a fact (Layer 4) is connected to a framework principle (Layer 1), and so on. Retrieval consults the layered stack for authority-ordered semantic access and the graph for relational access; the two are complementary.

Treating the graph as cross-cutting rather than as a sixth layer matters for two reasons. First, it preserves the meaning of layer order: the original paper's ordering is about authority and validity, and adding a sixth layer with neither property would muddy the semantics. Second, it accurately represents the operational role of the graph: it is a relational overlay, not a memory store with its own promotion pathway. The records it connects are governed by the pathways of their own layers; the relationships themselves inherit their status from their endpoints.

E.2 The Operational Substrate Is the Other Cross-Cutting Concern

The durable substrate introduced in I3 is similarly cross-cutting. Every layer has background work associated with it: identity-core versioning, working-memory expiry, outcome candidate generation, temporal-fact supersession checks, archive compression. In the original paper this work is implicit. A refined view of the architecture makes it explicit by identifying the operational substrate as a horizontal concern beneath the layered stack, on which all governance work runs. Just as the relational layer is consulted in addition to layered retrieval, the operational substrate executes in addition to request-thread work.

The combined picture is therefore: five vertically-stacked memory layers, ordered by authority and consulted in retrieval order; one cross-cutting relational layer connecting records across layers; one cross-

cutting operational substrate executing governance work for every layer. This is the smallest refinement that accommodates the integration patterns without restructuring the original architecture. It does not change the architecture's claims; it changes the diagram.

E.3 What the Refinement Does Not Change

Three properties of the original architecture remain unchanged. The layered structure of memory and the retrieval order across it remain as specified. The governance commitments—promotion through pathways P1–P5, validity states, supersession, and provenance—remain as specified. The evaluation dimensions E1–E6 remain the primary frame, with dual-frame retrieval-quality metrics added as a complementary signal rather than a replacement. The refinement adds detail to a previously-flat diagram; it does not move any of the architectural claims of the original paper.

F. Toward a Dual-Frame Evaluation Protocol

Section C.5 introduced dual-frame evaluation as an integration pattern. This section sketches the protocol in slightly more detail, because it is the part of the integration that most directly addresses the original paper's acknowledged limitation that it lacks empirical anchoring.

F.1 The Retrieval-Quality Frame

The retrieval-quality frame consists of three components: a domain-specific corpus, a set of evaluation queries with hand-labelled relevance judgments, and a suite of standard retrieval metrics. The corpus is the system's actual durable memory at a specified snapshot, exported in a form that can be re-indexed independently. The query set is constructed by domain experts and spans the system's scope: it should include representative queries from each scope partition, several classes of difficulty, and a sample of relational queries that exercise the graph layer. The relevance judgments are produced by domain experts who do not have access to the system's ranking signals, to avoid contamination.

The metrics are standard. Precision at k measures the fraction of the top-k results that are relevant; recall at k measures the fraction of the relevant results that appear in the top k; mean reciprocal rank measures the position of the first relevant result; normalised discounted cumulative gain accounts for ranking position and graded relevance. Reporting at a small number of k values (typically k=5 and k=10) is sufficient for most purposes. The metrics should be reported per scope partition as well as in aggregate, because aggregate metrics can hide scope-specific regressions.

The retrieval-quality frame is run on a schedule—ideally on every significant change to the retrieval stack—and the results are versioned alongside the code. A regression in any metric beyond a threshold triggers investigation. This frame does not evaluate continuity properties at all, and should not be expected to.

F.2 The Continuity Frame

The continuity frame consists of multi-session protocols evaluating each of the dimensions E1–E6. The protocols are necessarily more interpretive than retrieval metrics, because continuity properties are statements about behaviour over time rather than about single retrieval calls. The original paper provides task sketches for each dimension; instantiating those sketches in the system's actual domain is the work this frame requires.

Each protocol produces a structured judgment—either a quantitative score from a rubric or a qualitative annotated finding—rather than a single number. The judgments are made by domain experts reviewing the system's behaviour across the protocol's sessions. The cost is real: a single full continuity evaluation may take several days of reviewer time per evaluation cycle. The protocols are therefore run quarterly rather than continuously, and a subset is run on major releases.

Reporting from the continuity frame should preserve the per-dimension structure rather than collapsing into an aggregate. A system that scores well on E2 (temporal accuracy) and E3 (scope precision) but poorly on E1 (identity stability) has a specific problem with a specific remedy; aggregation obscures that information.

F.3 How the Two Frames Are Used Together

The two frames serve different purposes and should not be combined into a single score. The retrieval-quality frame is a development signal: it tells the engineering team whether a change to the retrieval stack improved or degraded retrieval. The continuity frame is a trustworthiness audit: it tells the system's users and stakeholders whether the system is behaving as a persistent expert should. A persistent expert system that publishes both is making a stronger commitment than one that publishes either alone.

There is a methodological subtlety worth flagging. The retrieval-quality frame can be gamed if the corpus and the queries are known to the team optimising the retrieval stack. The continuity frame can be gamed if the protocols become routine and reviewers rely on heuristic shortcuts. Both risks are real, and both are managed by the same discipline that applies to any benchmark: the corpus and protocols should be revised regularly, the relevance judgments and rubrics should be reviewed independently, and the published results should include enough methodological detail that readers can assess what was actually measured.

G. Open Questions and Future Work

The original paper closed with a set of open questions oriented to the architecture itself. This addendum suggests a complementary set, oriented to the integration between the architecture and the broader landscape of agent memory systems.

G.1 Where Is the Boundary Between Governance and Friction?

The production-agent class demonstrates that frictionless auto-accumulation is operationally powerful: systems that accumulate durable memory autonomously can produce substantial value with minimal human involvement. The persistent-expert architecture commits to governed promotion because the cost of wrong durable memory is high in its target domain. Between these two positions is a continuous space, and the placement of any particular system within that space is a design decision rather than a derived consequence.

What is not yet clear is whether the right placement depends primarily on domain (high-stakes domains warrant more governance; low-stakes domains warrant less) or primarily on temporal horizon (systems intended to persist longer warrant more governance; ephemeral systems warrant less). Empirical work that varies governance discipline within a fixed domain would help to answer this. The taxonomy proposed in Section B is a vocabulary for that question, not a resolution of it.

G.2 What Is the Right Unit of Identity?

The original paper treats identity as a first-class memory object, stored as a versioned record in the identity core. The production-agent class typically stores identity as configuration files (a system prompt, a behavioural specification, an access policy) that exist outside the memory system. These two stances reflect different assumptions about whether identity is something the system reasons over (memory) or something the system is constrained by (configuration).

Both stances are defensible, and the choice has architectural consequences. Treating identity as memory makes it queryable, comparable across versions, and subject to the same promotion pathways as other durable content. Treating identity as configuration makes it tighter, faster to enforce, and structurally less likely to drift. The persistent-expert architecture has committed to the memory stance for reasons internal to its design target; whether that stance generalises to other classes of agent memory system is an open question.

G.3 Does the Typed Relationship Vocabulary Generalise?

The relational layer introduced in I1 depends on a domain-specific relationship vocabulary. The production-agent class uses a vocabulary oriented to its target (people, companies, events, investments); the persistent-expert architecture for a coaching domain would use a different vocabulary oriented to its target (cohorts, components, outcomes, interventions). It is not yet clear whether there is a useful intermediate level of abstraction—generic across persistent expert systems but specialised to the class—or whether the vocabulary is inherently domain-bound.

If a class-level vocabulary exists, it would be a substantial methodological asset: persistent expert systems in different domains could share evaluation corpora, tooling, and best practices for relational extraction. If it does not exist, vocabulary design becomes part of the cost of building each persistent expert system,

and methodological transfer across domains becomes more limited. Investigating this question requires comparing the relationship vocabularies of multiple persistent expert systems in different domains, which is currently not possible because such systems are rare. The question becomes addressable as more persistent expert systems are built.

G.4 How Should Continuity Evaluation Itself Evolve?

The original paper proposes E1–E6 as the continuity-oriented evaluation dimensions. These dimensions reflect the failure modes F1–F6 the paper identified, and they are appropriate for the architecture as specified. As persistent expert systems are deployed in practice, additional failure modes will become visible that the current dimensions do not capture. The evaluation frame should be expected to evolve with the systems it evaluates.

Two candidate dimensions are worth flagging now, without proposing them as additions to the original frame. The first is robustness to scope drift: whether the system's declared scope and its operative scope remain aligned over time, or whether implicit scope expansion occurs through the accumulation of nominally on-scope content. The second is the cost of correction: how much work is required to revise a remembered item once it has been promoted, and whether the architecture's correction cost grows linearly, sub-linearly, or super-linearly with system age. Both dimensions are visible only over long deployment horizons and cannot be evaluated through short-term protocols. Their study is future work.

H. Conclusion

The original paper argued that persistent expert systems carry a continuity burden distinct from the recall problem that has organised most prior work on AI memory, and that meeting that burden requires a layered architecture with governed promotion, validity states, and continuity-oriented evaluation. This addendum has argued that the layered architecture can absorb five engineering patterns from a maturing class of production agent memory systems—typed relational extraction, compiled truth and timeline, durable operational substrate, tiered candidate triage, and dual-frame evaluation—without compromising its theoretical center, and that it must decline three other patterns from the same class for reasons that follow directly from the original paper's commitments.

The broader observation is that the boundary between governed and auto-accumulating memory is not a question of which approach is better, but of which design target a system is serving. The taxonomy proposed in Section B is intended as a vocabulary for that conversation. The integration patterns in Section C are intended as evidence that the layered architecture is generative—that it can absorb engineering from neighbouring systems while preserving the properties that distinguish it from them. The dual-frame evaluation protocol is intended to address the original paper's acknowledged limitation that its argument lacks empirical anchoring.

Persistent expert systems remain an emerging category. Their architecture, their governance, and their evaluation will continue to develop as more such systems are built and deployed. The original paper named the category and proposed an architecture for it; this addendum has integrated that proposal with the surrounding engineering landscape. The work of building, evaluating, and refining persistent experts in practice is ahead.

Note on References

This addendum draws on the references in the original paper. The production-agent class referenced in Section A is exemplified by GBrain (Tan, 2026), an open-source agent memory toolkit. Beyond that single citation, the patterns discussed are properties of a class of system rather than of any individual implementation, and are described accordingly.