Governed Corpus Evolution: Continuous Refinement of Knowledge for Frozen-Model Agents

Sviatoslav Zinevich

slavazinevich@gmail.com

August 2026

Abstract

A software project assisted by a large language model has a peculiar epistemic shape: its most capable component — the model — is frozen and rented, learns nothing between sessions, and holds no memory of the project. Everything the project *knows* must therefore live in text. This paper develops **Governed Corpus Evolution (GCE)**: a doctrine for running a typed, versioned document corpus as the persistent learning system of an agent whose generalizer cannot be trained. The thesis is threefold. First, the three functions that weight training bundles — storage, generalization, and behavior binding — separate cleanly under the frozen-model constraint, and each lands on a different substrate: the corpus stores, the model generalizes at read time while human judgment generalizes at write time, and mechanical gates bind what is bindable. Second, because the constraint removes the global optimizer, every persistent unit of knowledge must *locally* carry the whole learning loop; we derive a five-slot **entry contract** — activation, payload, warrant, enforcement split, lifecycle — and show that the familiar artifact families of an evolving project (rules, decision records, checkpoints, defect ledgers, mechanical gates, episodic notes) are one contract evaluated at different coordinates, not different kinds. Third, the write path must be **governed**: contradiction must come from structurally independent sources, adjudication must be unstaked, and permanent writes must pass a human gate — because the surveyed alternatives (autonomous, evaluator-gated, and agent-separated writers) exhibit measured collapse, contamination, and self-bias. The doctrine is distilled from a production system that ran the loop for several hundred sessions; it is argued as position and design theory, grounded in the memory-systems, context-engineering, and knowledge-engineering literatures, not as a benchmarked artifact.

1. Introduction

The practical unit of AI-assisted software development is the *session*: a context window is assembled, work happens, and the window is destroyed. The model that made the session capable retains nothing. It will not remember the architectural decision it helped make, the failure mode it helped diagnose, or the correction its user issued — and, decisively, its

weights are not available for the project to train. The model is *frozen and rented*: the project owns only the endpoints — what enters the context window, what leaves as action, and what gets written down afterward.

This situation is usually treated as a deficiency to be patched with a memory feature. We argue it should instead be treated as a design constraint with far-reaching consequences, and that the correct response is architectural: the project's document corpus — its instructions, decisions, rules, notes, and logs — *is* the learning system, and it can be run as one, deliberately, with the rigor that learning systems require. We name the resulting discipline **Governed Corpus Evolution (GCE)**: the corpus is *governed* at every write, and *evolution* — continuous refinement of what the project knows — is what a governed corpus does between writes. Since evolution proceeds only through writes, governing every write governs the trajectory.[1]

Thesis. When the generalizer is frozen, a project's knowledge can be run as a genuine learning system in text space if and only if three conditions hold:

- **T1 (Unbundling).** The three functions that weight training bundles — storage, generalization, behavior binding — are separated and reassigned: storage to a typed corpus, generalization to the model's in-context learning at read time *and* to human judgment at write time, behavior binding to mechanical enforcement plus disclosed residue.
- **T2 (The entry contract).** Every persistent unit of knowledge locally carries the whole learning loop — when it activates, what it caches, what warrants it, what enforces it, and how it retires — because nothing global remains to do so. The apparent diversity of knowledge-bearing artifacts in a mature project is this one contract evaluated at different coordinates.
- **T3 (Governance).** The write path is governed: beliefs are ledgered hypotheses exposed to structurally independent contradiction; the adjudicator is distinct from both proposer and contradictor; refinement is damped by recurrence evidence graded by blast radius; and a human gates every permanent write. This is not conservatism but a response to measurement: text-space credit assignment is unreliable, and every surveyed alternative write policy exhibits documented corruption modes.

Contributions. (i) A model of governed context — stores separated by decay profile, two-stage reads, event-sourced writes (§ 4). (ii) The entry contract as a unifying derivation, with worked instantiations and a failure mode named per missing slot (§ 4.5, § 6). (iii) The evolution machinery — a signal taxonomy, the hypothesis-ledger construct with its observed contradiction-source species, scheduled-judgment checkpoints, and a consolidation pipeline with depth-graded evidence bars — carried by the session loop and traced end-to-end through one lesson's full lifecycle (§ 5). (iv) An account of how the doctrine instantiates

across the artifact families and agent roles of a working project, together with the genesis sequence by which a project bootstraps the loop (§ 6). (v) A positioning of GCE against seven neighboring families of systems and practices, and an honest failure surface (§§ 2, 7, 8).

Method and evidentiary status. This is a doctrine paper in the tradition of position and systems-design papers rather than an empirical study. Its claims arise from one production source: a solo-architect monorepo developed with LLM agents across several hundred working sessions, whose process corpus grew, under continuous adversarial review, into the system described here. Where the paper cites quantitative results they are the literature's, not ours; where it makes claims about the origin system they are reports of a single deployment, and § 7 states the resulting validity limits plainly — including the one this paper cannot escape: the strongest test of a doctrine extracted from one system is a second, independent adoption.

---

[1] The doctrine has circulated internally under two bindings — *governed context* evolution (context: what is assembled per session) and *governed corpus* evolution (corpus: the thing that persists and evolves). We use "corpus" here — it names the thing that actually persists and evolves — while keeping "context" for what is assembled per session, in continuity with the context-engineering literature; the constructs are identical.

2. Related systems and lineage

GCE sits at the intersection of seven literatures. We take them in order of proximity, and for each we name the specific inheritance and the specific divergence.

2.1 Self-evolving context and experience-driven agents

Karpathy (2025) named the missing paradigm *system-prompt learning*: an LLM should accumulate explicit, textual problem-solving knowledge — "a book for itself" — edited rather than trained, in a loop that resembles reinforcement learning in setup but with edits in place of gradient descent. A rapidly growing family instantiates the idea with the model as its own librarian: Reflexion converts failure feedback into verbal self-corrections (Shinn et al., 2023); Voyager accretes an executable skill library under a frozen GPT-4 (Wang et al., 2023); ExpeL extracts insights and rules from pooled experience with vote-count lifecycles (Zhao et al., 2024); Generative Agents stream observations into a retrieval- scored memory with importance-triggered reflection (Park et al., 2023); Agent Workflow Memory induces reusable workflows online, gated by an automated evaluator (Wang et al., 2024); Dynamic Cheatsheet curates a test-time memory of strategies (Suzgun et al., 2025); ACE frames the evolving context itself as the self-improvement substrate, with delta-updates by a curator model (ACE, 2025); Training-Free GRPO ports the RL update shape into pure text-space

experience libraries (2025); ArcMemo stores concept-level abstractions rather than instances (2025). Surveys now organize the space (Zhang et al., 2024; Gao et al., 2025; Fang, Peng et al., 2025), with the most recent framing agent memory's own history as an evolution from storage through reflection to cross-trajectory abstraction (Luo et al., 2026).

GCE shares this genus — it is system-prompt learning at project scale — and diverges on exactly one axis: **who writes**. The self-evolving field lets the model author its own memory and then, in its own literature, measures the consequences: context collapse (an 18,282-token accumulated context self-rewritten to 122 tokens in one step, landing below the no-memory baseline; ACE, 2025), brevity bias, memory dilution and retrieval pollution as forgetting relocates from weights into retrieval (When Continual Learning Moves to Memory, 2026), and self-bias amplification when models evaluate their own outputs (Xu et al., 2024). GCE's write path is constitutionally removed from the model: the machine enumerates, drafts, proposes, and audits; a human authors and gates (§ 3). Notably, the origin system's countermeasures — append-only piles, delta-not-rewrite, human-gated writes — predate the papers that measured the failures they prevent: convergent evolution from opposite directions, which we read as the strongest available validation of both (§ 8.2).

2.2 Agent memory systems

An industry has formed around persistent memory for LLM agents. MemGPT introduced OS-style tiered context with self-directed paging (Packer et al., 2023), and its successor Letta added an *agent-separated* write policy: a background "sleep-time" agent holds the memory-edit tools that the acting agent lacks (Letta, 2025). Mem0 runs LLM-driven fact extraction with add/update/delete resolution (Chhikara et al., 2025); Zep/Graphiti maintains a bi-temporal knowledge graph in which contradiction invalidates edges rather than deleting them (Rasmussen et al., 2025); A-MEM links LLM-authored atomic notes Zettelkasten-style and lets new memories rewrite old ones' attributes (Xu et al., 2025); MemoryBank decays memories on an Ebbinghaus-inspired schedule (Zhong et al., 2023); MemOS unifies memory classes under a governed "MemCube" abstraction with provenance and versioning — governance at the system level, not the human level (Li et al., 2025). Consumer products (ChatGPT memory; Claude memory) write automatically and offer *post-hoc* audit: view, edit, delete — never pre-write approval.

Across this landscape a write-policy spectrum is visible: fully autonomous LLM writes → evaluator-gated writes (AWM online) → agent-role-separated writes (Letta) → post-hoc human audit (consumer memory) → human-authored instruction files (§ 2.3). To our knowledge, **no major production memory system pre-gates memory writes on human approval by default**; diff-then- approve appears only as a proposed wire-format channel

(memorywire, 2026), as an emerging hand-rolled production pattern, and in one local-first event-sourced prototype that gates agent *actions* off memory rather than memory writes by a human (PROJECTMEM, 2026). GCE occupies the unclaimed end of the spectrum and argues for it (§ 3, § 8).

Two 2026 results independently support GCE's structural bets from within this field: typing pays — heterogeneous memories sharing one retrieval space contaminate each other, and explicit functional-role typing at write time restored reliability (+28% in MemGuard's evaluation; Ha et al., 2026) — and write-time sophistication is overrated relative to retrieval: raw chunk storage matched or beat LLM extraction pipelines while retrieval method spanned twenty accuracy points (Yuan, Su & Yao, 2026). Both echo GCE's allocations: one store per decay profile (§ 4.2), and design attention concentrated on the read path's mechanical stage (§ 4.3). Meanwhile the field's own quality literature has turned longitudinal and adversarial: memory-induced safety violations rise monotonically with exposure length across architectures (Al-Tawaha et al., 2026); memory stores are demonstrated poisoning targets at trivial poison rates (MINJA, 2025; AgentPoison, 2024), now codified as OWASP's agentic risk ASI06 — findings that convert "who may write to memory?" from a philosophical question into a security one (see also Willison, 2025, on the lethal trifecta).

2.3 Context engineering and knowledge-as-code practice

Practitioner vocabulary converged in mid-2025 on *context engineering* — "providing all the context for the task to be plausibly solvable" (Lütke, 2025; Willison, 2025a), now surveyed academically (Mei et al., 2025) and codified by vendors (Anthropic, 2025a). The artifact ecosystem is rich: repo-root instruction files (AGENTS.md, adopted by tens of thousands of projects and stewarded by a foundation; CLAUDE.md with tiered scopes and imports), matched-loading rule systems (Cursor's four rule types; Kiro's steering files with four inclusion modes; path-scoped rule files), progressive-disclosure skill packages (Anthropic, 2025b), and spec-driven pipelines with human-ratified constitutions (GitHub Spec Kit, 2025; Kiro, 2025). Two facts from this practice matter to our argument. First, vendors *independently converged* on the same three-way loading taxonomy — always loaded, matched, on-demand — that GCE derives from first principles as stage-one attention ownership (§ 4.3); convergence under competition is evidence the taxonomy is real. Second, the practice literature knows its failure mode: instruction files degrade with length ("context rot" is now measured — performance falls non-uniformly with input tokens even on simple tasks; Hong, Troynikov & Huber, 2025 — and vendor guidance caps instruction files and warns that bloat costs adherence), and promotion-on-evidence between tiers exists only as informal advice ("add the rule when the same mistake happens twice") — nowhere, in our sur-

vey, as a formal governed mechanism. GCE supplies exactly the missing machinery: capacity-capped tiers with forced eviction, and an evidence- graded promotion pipeline (§ 5.4).

2.4 Retrieval-augmented generation

GCE's read path is deliberately RAG-shaped — cheap recall, precise selection, generation — so the divergence is not retrieval mechanics but write-side lifecycle: RAG stores chunks verbatim and evaluates retrieval offline; it never promotes, never abstracts on the way up, never tiers by decay profile, never credit-assigns a write. GCE additionally refuses the embedder: its retrieval keys are symbolic and legible on purpose, because the embedding model is as rented as the generator, and legible keys survive a model swap and transfer better under distribution shift (§ 4.3). Graph-and-memory hybrids (HippoRAG, 2024; Zep, 2025) mechanize richer retrieval over auto-written stores; GCE spends that budget on governed writes over simpler retrieval.

2.5 In-context learning versus weights

The temptation to call a curated corpus "fine-tuning by other means" fails on the mechanism: the ICL≈gradient-descent equivalence holds only in toy regimes (von Oswald et al., 2022; Dai et al., 2022) and is refuted for real models, where order-sensitivity makes the two provably different functions (Shen et al., 2024), and in-context and in-weight learning appear as separate, rivalrous circuits (Chan et al., 2022; Singh et al., 2023). What survives is the *slot* claim GCE builds on: for factual and procedural payloads, context injection beats fine-tuning outright (Ovadia et al., 2024), and converting context into parameters is a real, lossy research program of its own (Cartridges, 2025) — so a project that keeps its knowledge in context keeps it reversible, attributable, and live across model swaps.

2.6 Automated prompt and text optimization

TextGrad, GEPA, OPRO, ProTeGi, and semantic backpropagation close the text-update loop with the model inside it (Yuksekgonul et al., 2024; Agrawal et al., 2025; Yang et al., 2023; Pryzant et al., 2023; 2024). GCE refuses the closure at the one point where it matters: attribution. Text-space credit assignment is measured weak — the best automated methods locate the culpable component 53.5% of the time and the decisive error step 14.2% of the time, some below random (Zhang et al., 2025) — and even the gradient analogy's proponents concede its explanatory gaps (2025). GCE therefore runs attribution through human-plus-strong-model judgment, damped by recurrence bars (§ 5.4); the optimizer analogy survives only as governance polarity — the human as the single, deliberate, low-frequency backward pass (§ 3).

2.7 Symbolic AI and knowledge engineering

The deepest lineage. Truth maintenance systems are the classic ancestor of GCE's ledgered hypotheses: Doyle's TMS stored every belief with explicit justifications and made retraction a first-class propagating operation (Doyle, 1979), and de Kleer's ATMS labeled beliefs with the assumption sets under which they hold (de Kleer, 1986) — but propagation was automatic, where GCE deliberately inserts adjudication between contradiction and retraction. AGM belief revision axiomatized minimal change under an entrenchment ordering (Alchourrón, Gärdenfors & Makinson, 1985) — a normative ancestor of tiering that is silent on who decides entrenchment; GCE's answer is § 3's asymmetry. Case-based reasoning made retention a deliberate pipeline stage and then discovered the *utility problem*: an ever-growing experience store degrades, forcing competence-preserving deletion policies (Aamodt & Plaza, 1994; Smyth & Keane, 1995; Leake & Wilson, 1998) — the ancestor of GCE's retirement legs. Soar's chunking is the canonical promotion-raises-abstraction mechanism, compiling solved-goal traces into rules — ungated, with a known wrong-chunk problem (Laird, Rosenbloom & Newell, 1986); ACT-R's declarative/procedural split with recency-frequency decay is the ancestor of decay-profiled stores (Anderson & Schooler, 1991) — a lineage the language-agent literature has itself reconnected to (Sumers et al., 2023). Knowledge engineering named the human bottleneck (Feigenbaum, 1977), and Cyc is its cautionary monument: decades of hand-curated assertions demonstrating that human gating *alone* — without recurrence thresholds, decay profiles, or retirement — yields an unbounded maintenance liability (Lenat, 1995). Blackboard architectures had independent knowledge sources posting competing hypotheses arbitrated by a scheduler (Erman et al., 1980) — within one episode, with a staked arbiter. Wikipedia is the largest natural experiment in a self-governing versioned corpus, with promotion tiers (essay → guideline → policy) gated on demonstrated consensus (Forte & Bruckman, 2008; Butler et al., 2008) — and with the measured failure GCE must also answer: rule mass without decay ossifies and repels contributors (Halfaker et al., 2013).

From organizational science GCE inherits framing rather than mechanism: double-loop learning for a corpus that revises its own rules for revising (Argyris, 1977); externalization of tacit knowledge as the lossy, load-bearing step (Nonaka, 1994); typed retention bins with distinct retrieval characteristics (Walsh & Ungson, 1991); and the transactive- memory contrast — a human–agent dyad cannot rely on "knowing who knows what" because the agent's directory resets every session, so the corpus must *be* the directory (Wegner, 1987). From engineering practice it inherits working precedents: architecture decision records with immutable numbering and supersedure pointers (Nygard, 2011; adr.github.io; ThoughtWorks, 2017); the IETF's append-only RFC lifecycle with `Updates:` / `Obsoletes:` pointers and maturity tiers (Bradner, 1996); documentation typed

by reader need (Procida, 2020); pre-registration as the freeze that makes scoring honest (Nosek et al., 2018); blameless postmortems as typed failure records (Beyer et al., 2016); checklists as the empirical case that mechanizing *known* discipline beats expert recall (Haynes et al., 2009; Gawande, 2009); bounded bets with a default-to-termination circuit breaker (Singer, 2019); and the design- rationale literature, whose central negative result — the *capture problem*: recording rationale costs the designer at exactly the moment they are least willing to pay (Conklin & Begeman, 1988; Lee, 1997) — is the strongest argument that in the LLM era rationale capture should be agent-drafted and human-gated, since drafting is precisely what became cheap.

3. The root constraint and the doctrine it forces

The model's weights are not ours to change. The map is rented; only the endpoints are owned — what enters context, what leaves as action, what gets written down. Five doctrine positions follow, each turned from limitation to advantage:

1. **Own the endpoints; treat the model as a fixed oracle.** No learned routing, no auto-judged applicability, no tuned embedder in the retrieval path. Legible symbolic keys transfer under novelty and survive a model swap; a latent index does neither.
2. **Cache judgment in legible text, never in weights.** Text can be diffed, attributed, audited, and rolled back; nothing baked into a parameter can. Reversibility and attribution are non-negotiable when the substrate is rented.
3. **Amortize the human's judgment, not the model's inference.** Each hard question surfaces once, at a scheduled checkpoint, and its answer crystallizes into a store with a lifecycle. The standing tax is the lifecycle machinery that keeps the cache honest.
4. **Place every judgment on the deepest available key** — the duty as narrow as is true, the hook as broad as is predictive. Placement sets steering leverage and drift exposure on the same axis.
5. **Behavioral feedback is forced, not chosen.** Activations and embeddings are invisible; behavior is the only channel. Corollary: *acted-upon* beats *technically-correct* as the quality instrument, and catch-rate is never a target. And within the behavioral channel, one signal is sovereign: the human's correction — the steer — is the sole durable *reward*. The world contradicts and counters accumulate, but only the steer assigns value; the machinery for capturing it is therefore a first-class, independently tunable component of the system (§ 5.1).

The species differentiator against the self-evolving field is **governance**, and one asymmetry inside it shapes everything downstream: **promotion is owner-performed — always.**

Consolidation — an observation becoming a decision, a decision becoming a rule, a rule's mechanical floor becoming a gate — is never executed by the agent and never by the world, for two different, load-bearing reasons. The agent is excluded because self-performed promotion is self-referential: the context that emitted a hypothesis would be ratifying it — the measured self-bias and collapse failure of the auto-writing field (§ 2.1). The world is excluded because the world cannot write — it can only contradict: outcomes ground claims but do not decide what the lesson is, and deriving structure directly from outcomes is the noisy-attribution trap (§ 2.6). In borrowed training vocabulary — with the standing caution that the analogy supplies words, never mechanism (§ 8.3) — sessions are the forward pass, the world supplies the loss signal, agents compute *proposed* updates with evidence attached, and **the human is the optimizer step**: one deliberate, low-frequency backward pass that alone applies updates to the store. The throughput bottleneck this creates is accepted — it is the point — and it is why proposals must arrive self-contained, with all five contract slots filled (§ 4.5), so each human judgment surfaces once, cheaply, at the moment it can still change the outcome.

4. Modeling governed context: the statics

4.1 The three functions of training, unbundled

Weight training bundles three separable functions; GCE unbundles them, and the unbundling is where every borrowed training word both fits and breaks:

1. **Storage** lives in the corpus — natively better than weights for this payload: reversible, attributable, live-next-session with zero retraining, and, per the knowledge-injection literature, outright stronger than fine-tuning for factual and procedural content (Ovadia et al., 2024).
2. **Generalization** splits in two. It is *outsourced to the frozen model's in-context learning at read time* — the corpus curates, the model generalizes. And it is *performed by human judgment at write time*, in the abstraction step of promotion: restating an instance object-decoupled, at transferable altitude, before it moves up a tier. This step is load-bearing, not stylistic: stored raw instances produce negative transfer that flips positive when stored as abstracted insights (2026; corroborated at concept level by ArcMemo, 2025), and its symbolic ancestor is Soar's chunking (§ 2.7). Promotion that copies instead of abstracting builds a recall cache, not a learning system.
3. **Behavior binding** is the weakest link of every context system — a rule acts only if retrieved AND applied. GCE binds mechanically where binding is bindable (gates enforcing the machine-checkable floor of each rule) and directs attention to the rest by disclo-

sure: every rule states, in its own text, what its enforcement does *not* check. The field mostly hopes.

4.2 One store per decay profile; never one governance

The corpus is not one pile. Different knowledge rots at different rates, is recalled at different moments, and fails differently — so it belongs in stores with different recall modes, capacity policies, and forgetting mechanisms. The origin system stabilized on six:

Store class	Holds	Recall mode	Forgetting
Constitutional	The always-loaded rule tier	Loaded every session	**Hard cap with forced eviction** — the system's only fixed-width layer
Rule	Recurring conditional duties (condition → duty)	Routed associatively by declared work-shape	Demotion review; probation; tombstones
Case	Settled point decisions that constrain the future	Full scan of one-line summaries + native matching	Status flip (superseded / moot); never deleted
Episodic	Not-yet-evidence: observations, felt friction, typed defect records	Trigger-fired + periodic passes	The only delete-with-pointer piles — promotion or obsolescence empties them
Procedural	Gates, skills, generators	Mechanical execution; progressive disclosure	Retirement on supersession
Trace	Session logs — the event record	Header at boot; projections on demand	None — everything else regenerates from it

The claim is not the roster — an adopting project grows its own — but the separation law: **one store per decay profile; never one governance.** Its independent support is now measured: heterogeneous memories sharing one retrieval space contaminate each other, and explicit typing at write time restores reliability (Ha et al., 2026); its organizational ancestor is the typed retention-bin taxonomy (Walsh & Ungson, 1991); and its cautionary contrapositive is Wikipedia's undifferentiated rule mass (Halfaker et al., 2013).

The hard cap on the constitutional tier deserves its own sentence, because it is the most frequently resisted rule in practice: the always-loaded tier carries a hard capacity cap on both item count and total mass, and adding a rule means evicting one. Soft budgets fail — this is both the origin system's scar and the practice literature's convergent warning (instruction-file bloat costs adherence; context performance degrades non-uniformly with length even on simple tasks; Hong et al., 2025). Exactly one fixed-width layer exists in the system, and the cap is what forces the ranking conversation that a growing pile silently avoids.

4.3 Reads are two-stage attention

Every read surface in the system is one pattern: **a mechanical, legible, hard pre-attention stage selects what enters context** — routing vocabularies, trigger predicates, per-document section maps, progressive disclosure — **and the model's native attention does the conditioning.** All design content lives in stage one, the only stage that is ours.

Which stage owns which selection tracks three variables: corpus size × per-item token weight × miss cost. Heavy, numerous, high-miss-cost entries (the rule store) get routed activation with declared firing shapes; cheap one-line summaries with high miss cost (the case store) get bare full-scan with the model's attention doing the matching; heavy documents get section maps and partial reads; procedures get name-and-description tiers with bodies on demand. The placement heuristic: **move selection into stage one only when items are too heavy to scan; keep stage-one keys legible, because there is no gradient with which to repair an illegible key space.**

Two external facts triangulate this section. The tool industry, under competition, independently converged on the same taxonomy — always-loaded instruction files, glob- and description-matched rules, on-demand progressive-disclosure skills (§ 2.3) — which we read as convergent discovery of the stage-one design space. And the memory literature's 2026 diagnosis — retrieval method dominates write-time sophistication by an order of magnitude in accuracy spread (Yuan, Su & Yao, 2026) — locates the leverage exactly where this section spends it.

4.4 Writes are event-sourced

Single canonical source; every index a regenerated projection; IDs reserve-once, immutable, gap-tolerant; retirements tombstone with successor pointers; refinement is a new record or a status flip, never a rewrite of evidence. A hand-maintained parallel surface is drift by construction. The precedents are exact: RFC immutability with `Updates:` / `Obsoletes:` pointers (Bradner, 1996) and ADR practice's never-reuse-numbers, never-delete-superseded discipline (Nygard, 2011).

4.5 The entry contract: the pentad

The section above describes stores; this one derives what every entry in any store must carry — the paper's central theoretical claim.

In a weight-trained system, five functions are *global* properties of the training loop: attention learns when knowledge activates; the loss warrants it; the weights bind it; continued training refines and retires it. The root constraint removes the optimizer — so nothing glob-

al will ever activate, warrant, bind, or retire a piece of cached judgment. Therefore **every persistent entry must locally self-carry the whole learning loop, in legible text.** Every governed construct converges on one five-slot contract — the **entry contract**, informally the *pentad*:

1. **Activation** — when does this enter context, or fire?
2. **Payload** — the cached judgment, at transferable altitude.
3. **Warrant** — what makes it correct: anchors, truth-maker, adjudication — causally independent of the entry itself.
4. **Enforcement split** — the mechanical floor plus the disclosed residue.
5. **Lifecycle** — the named consumer plus the retirement leg.

The contract also has a derivation from above, and stating it requires stating the base this paper has so far carried only in prose: the doctrine's corpus-wide commitments, accumulated through §§ 3–5, as an explicit invariant set (the last four anticipate § 5's machinery):

- **I1** — One store per decay profile; never one governance.
- **I2** — Writes are human-gated: the machine enumerates, proposes, and audits — it never authors.
- **I3** — Single canonical source; every index a regenerated projection; immutable IDs with tombstones. A verifier's independence from its subject is causal, not spatial.
- **I4** — Progressive disclosure everywhere; stage-one keys stay legible.
- **I5** — No record without a standing consumer.
- **I6** — Promotion must raise abstraction, never tier-copy an instance.
- **I7** — Every entry threshold pairs with a retirement leg: counts nominate, competence decides, killer-items are exempt regardless of count.
- **I8** — The always-loaded tier carries a hard cap with forced eviction; exactly one fixed-width layer exists.
- **I9** — Feedback is behavioral only; catch-rate is never a target.
- **I10** — Every rule, gate, and checkpoint discloses what its enforcement does not check.
- **I11** — Noise-filter the error stream before updating process: reducible failures update authoring; irreducible ones update detection.
- **I12** — External signals outrank self-graded ones; self-graded labels never drive permanent structure alone.
- **I13** — Every held belief is a marked, anchored hypothesis; suspected and verified never mix registers.

- **I14** — Every belief class that matters has a structurally independent contradiction source, adjudicated by a party distinct from both proposer and contradictor.
- **I15** — Contradictions land append-only in the domain that held the failed belief: one owning ledger per domain.
- **I16** — The instruments are instrumented: attacker precision tracked, misses recorded, vocabularies blind-coded, floors honest about observed ≠ occurrence.

The pentad is the **per-entry projection of this set**: the invariants are stated corpus-wide; the pentad is what they demand of each single entry. The mapping is exact — activation is I4/I8's per-entry face; payload is I6's; warrant is I3/I12–I14's; the enforcement split is I10's; lifecycle is I5/I7's. Necessity follows from the removed-optimizer derivation above. Sufficiency — are five slots *enough*? — is held as an empirical claim under the doctrine's own escape discipline: if weight training supplied a sixth global function the contract failed to localize, some organ would eventually be born needing a sixth slot; across the origin system's organ births none has, and the claim stays open in exactly the posture § 8.4 prescribes.

Three properties make the contract more than a checklist. First, it is a *derivation*, not a convention: in the origin system, organs kept being born five-slot-shaped with nobody mandating it — convergent evolution inside one constraint field. Second, **slot fillings are coordinates, not choices**: artifact families differ because each evaluates the contract at its own position on a handful of axes the doctrine owns — recall economics, truth-maker, bindability, decay profile, moment in the loop — which § 6 works through family by family. Third, the contract doubles as an instrument, because **each missing slot names a known failure mode**: no activation → the structural zero (stored but never retrieved — the new forgetting); no independent warrant → the self-warranting verifier (§ 5.2); no floor disclosure → complacency about what enforcement does not check; no lifecycle consumer → the graveyard, or entry-without-retirement (the documented failure of every recurrence-based memory, from CBR's utility problem to ExpeL's evict-at-zero rationale); payload un-abstracted → the recall cache. An organ proposal missing a slot names its own failure row — which turns "design a new knowledge artifact" into "fill five slots and defend each."

5. Evolution: the refinement machinery

The statics store and recall knowledge; this section is how the system *finds out it is wrong* and refines — the evolution in the name.

5.1 Signals, the steer, and the detection matrix

Four signal classes, deliberately not conflated:

- **The steer** — the human's correction, the reward channel. An n=1 supervised specification edit that also performs credit assignment: it decides *which artifact* absorbs the lesson.
- **The world** — the environment channel: failing tests, live behavior, incidents, gate fires. Grounds claims; rewards nothing. Conflating observation with reward is a category error the RL framing invites.
- **Recurrence counters** — the accumulator: nothing structural updates on one instance (§ 5.4).
- **Self-noticed signal** — lowest trust: self-refinement amplifies self-bias (Xu et al., 2024), so self-graded labels are trust-tiered and never sole drivers of permanent structure.

The taxonomy is not flat: **the steer is the sole durable reward channel, and everything else is either evidence or bookkeeping.** The world can falsify a belief but cannot say what the lesson is worth; counters can nominate but not decide; self-noticed signal cannot be trusted alone. Every unit of value the corpus will ever encode enters through a steer — which makes **steer capture an independent, first-class design concern**: apparatus whose sole purpose is to maximize how much of each steer's content survives into the record, correctly credited. Its instruments are recognizable GCE machinery turned to this one purpose: typed steer records written at the moment of correction (before the context that understood the correction is destroyed); a classification vocabulary over steer kinds, grown by the staged discipline of § 5.6 — whose escape hatch is here not hygiene but capture apparatus, since a steer that fits no class is precisely the one that must not be lossily forced into one; and the reflect step's scheduled steer moment (§ 5.7). Like every instrument, the capture apparatus is itself instrumented and tunable (I16). One polarity guards the concept: the quantity to maximize is *yield per steer* — content captured, credit assigned — never steer *count*. Soliciting corrections manufactures reward the way targeting catch-rate manufactures catches (I9); in a healthy system steers become rarer, for the reason the matrix below makes visible.

Where do steers come from? Every catch-or-miss event in the system falls in a two-by-two — the system's own apparatus (routing, latches, gates, examiners) on one axis, the human on the other — and the four cells have sharply different meanings:

	Human catches	Human does not catch
System catches	**Redundant catch.** The mechanical catch made the steer unnecessary — confirmation that a minted duty now fires on its own, and *demotion-side* evidence: mass here nominates coverage migration	**The amortization working.** Zero marginal human cost; the majority cell by design, and the cell to which a healthy system migrates its mass
System misses	**The steer.** Reward and credit assignment in one — and simultaneously a miss report against the system itself: every steer carries a second, free payload ("why didn't the apparatus fire?") that feeds the slot analysis of § 5.4	**The true miss** — the hardest class. Invisible until the world votes; reaches the system only through the reality species (§ 5.3) as a defect, and is repaired *backwards in time*: judgment-performed backtrace from the manifested harm to the belief that failed, with the direction-paired archaeology closing the causal ambiguity. Counts here are floors, permanently (I16)

The matrix is the system's health trajectory made legible: maturation is mass migrating from the steer cell into the system-catches cells, with the true-miss cell bounded — never measured — from the reality stream. It is also the steer's scarcity argument in one picture: the bottom-left cell is the only durable reward source the system has, each of its events is unrepeatable (the same human, shown the same miss twice, is a process failure, not a second datum), and that is why the capture apparatus above is worth building as its own concern.

5.2 Hypothesis ledgers

Every consequential belief the system commits to text — a plan's assumption, a diagnosis, a test expectation, a prediction, a decision, a rule, a class label — is a **hypothesis with an anchor**, never a bare assertion: the label is a hypothesis, the anchored instance is the datum; confidence is marked, never implied; suspected and verified never mix registers.

A **hypothesis ledger** is a typed disagreement store — the five-slot tuple:

〈 claim · contradiction source · adjudicator · verdict vocabulary · consolidation consumer 〉

The claim is frozen verbatim with its anchors (contradiction against a drifted paraphrase is contradiction against a strawman). The contradiction source is *structurally independent* — and independence is causal, not spatial: the source must be load-bearing for something other than this measurement, so its disagreement is evidence rather than echo. The adjudicator is distinct from both proposer and contradictor. The verdict vocabulary is closed with an honest escape. The consumer is the named periodic pass that reads recurrence and routes it — a ledger with no consumer is a graveyard. Records are append-only with immutable IDs;

the system's history of being wrong is its most valuable training corpus. The classic ancestor is the TMS belief node with its justifications (Doyle, 1979); the difference is the deliberately inserted judgment seam between contradiction and retraction.

The tuple encodes a constitutional principle: **proposer, contradictor, adjudicator, and committer are distinct parties**, and each collapse of two roles into one is a named failure. Proposer = contradictor → no contradiction is ever generated; hypotheses survive their author's scrutiny by default, and a failed refutation by an independent attacker is worth more than any number of author confirmations. Contradictor = adjudicator → the attacker grades its own attack, so every contradiction is "confirmed"; the attacker's precision must be tracked by someone else. Proposer = adjudicator → the *self-warranting verifier*: the disagreeing party decides the disagreement (a test author whose test unexpectedly fails must quarantine and report, not decide whether test or implementation is wrong). Adjudicator = committer, unattended → the ungoverned auto-write loop, with the field's measured collapse and self-bias (§ 2.1). The human gate on permanent writes is the last, non-collapsible seam — and, after the 2025–2026 poisoning results turned memory stores into attack surfaces (§ 2.2), it is a security control as much as an epistemic one.

5.3 The contradiction-source species

A boundary must be drawn before the taxonomy: **what the doctrine necessitates is the family, not the roster.** The hypothesis-ledger construct of § 5.2 — the five-slot tuple and the separation of powers — is entailed by the thesis; *which* ledgers a project runs is not. Ledgers differ by where their contradiction comes from, and the origin system's set stabilized at six **species**, presented here as an observed taxonomy and a menu for implementers — each species independently adoptable, none individually required for the doctrine to be active:

Species	Contradiction source	Distinctive law
Attack	Manufactured — an adversarial examiner with no authoring stake, dispatched against a proposed artifact *before commitment*, phrased to refute	Targets the artifact's own claim list; survived-with-attack-named is confirmation value; the attacker's misses are recorded retroactively; the highest-value attack class is the premise kill
Collision	Emergent — two honest parties independently derive one contract (canonically: a test author deriving expectations blind from committed artifacts) and disagree	Nobody attacked anybody — the disagreement is the datum; the verdict assigns fault among **three** parties: derivation wrong, implementation wrong, or *the contract is open* — an undecided fork surfaced, routing to specification work

Species	Contradiction source	Distinctive law
Forecast	The future — predictions and kill-criteria frozen at episode entry, scored at boundaries, never edited	Pre-registration transposed into planning (Nosek et al., 2018); the only species that refines the *predictor* rather than the corpus — accumulated scorecards become calibration priors about the planning faculty
Reality	The world, unscheduled — defects that manifest	Counts are floors, never rates (observed-zero ≠ zero); attribution runs backward through judgment at a periodic pass; paired archaeology streams (fixes that landed before codification; codifications that forced fixes) close the causality ambiguity either alone carries
Currency	Time and drift — the standing hypothesis "this is still true"	Declared at write time: a falsification list plus revisit triggers make staleness *scheduled* rather than discovered; retirement keys on mootness or coverage migration, never on low salience — the rarely-binding entry can be the killer-item; watch referents, never aliases
Coding	A blind second coder — labels versus labels over shared anchors	The countermeasure to self-emitted-taxonomy bias: an independent party classifies raw anchored instances *without seeing* the candidate labels; agreement ratifies a class, disagreement keeps the boundary open

The table carries an internal gradient that matters to an adopter. Two species are **passive** — their contradiction sources exist unbidden. The world votes whether or not a reality ledger is kept (every project has defects; the only choice is whether they become typed records), and time passes whether or not currency is declared (the currency species is arguably entailed by the entry contract itself — it is the lifecycle slot's own contradiction channel, run deliberately). The other four are **active**: their contradiction is *manufactured* — an examiner dispatched, a blind derivation staffed, a prediction frozen, a second coder run — and each is an investment in independence machinery, made when a project can afford the seams and the source exists. Admission is by test, not by template: a new species (or pile) is warranted only when a belief domain no existing store's lifecycle owns meets a structurally independent contradiction source, an unstaked adjudicator, and a named consolidation consumer — and refusal is a first-class outcome. The doctrine is running as soon as one species runs with all five tuple slots filled; the origin's ordering — passive species first, active species as independence machinery became affordable — is § 6.5's genesis rule, not a law. And the set is closed-with-escape (§ 8.4): a seventh species claimed does not embarrass the table; it re-derives it.

5.4 The slow loop: consolidation as a slot-factored backward pass

The fast loop — the session, § 5.7 — generates signal; this section names its counterpart with equal explicitness. The **slow loop** is the set of scheduled passes that consume accumulated signal and move knowledge between tiers, and the passes are few and nameable: the **episodic pass**, which walks the observation piles grouping by shape — trigger-fired when a re-check condition matches the session's work, periodic when a pile grows heavy; judgment-bounded, never cadence-bound; the **classification passes**, which blind-code accumulated ledger records against their raw anchors (§ 5.3, coding species); the **demotion reviews**, which read use-time telemetry; and the boundary ceremonies of § 6.3. Consolidation runs at these passes and nowhere else — which is to say, where the human is present: the governance property that distinguishes this trigger family from the field's idle-triggered and accumulation-triggered designs.

Recurrence — never a single record — drives structural change, and three disciplines — together, the refinement law — govern every pass.

Noise-filter before updating process. Every failure is classified irreducible-at-the-time (no authoring duty could have prevented it — do not update the authoring process on it) versus reducible (a duty was missed — route it). This is the aleatoric/epistemic split performed by judgment, and it prevents overfitting process to one-off failures. Irreducible recurrences are not discarded: they tune the *detection* side — when to dispatch the attack, what it leads with — never the authoring side.

Route before minting. A ratified recurrence lands at the *cheapest sufficient home*, walked in order: an edit to an existing template → a row in an existing duty table → a rule enrollment → a hook edit on an existing checkpoint → a new checkpoint (only on a genuinely new authority join) → a mechanical gate (only where fully mechanical). Refusal is a first-class outcome: every standing mechanism taxes every future session, and the ladder prices that at admission. (The strongest institutional precedents for checkpoint discipline are *refusals* to mandate checkpoints.)

Bars grade with blast radius. A cheap episodic note takes one noticing. A promotion candidate takes two anchored recurrences. A rule takes an observed recurrence *plus* human approval. A mechanical gate takes two instances of silent failure. The always-loaded constitutional tier takes repeated evidence at scale plus a forced eviction under its hard cap. Deeper, wider-reaching layers update slower — not bureaucracy but the damper an unreliable attribution channel requires (§ 2.6). The specific bar values are house convention, honestly labeled as such; the field's equivalents gate on vote counts and importance sums (ExpeL,

2024; Park et al., 2023) and derive their currencies no better. What is principled is the shape: counts nominate, the human decides, and promotion must raise abstraction (§ 4.1).

The slot-factored backward pass

What remains is the question the disciplines above presuppose: given accumulated evidence against an entry, *what kind* of refinement is needed? Weight training answers its version by factorization: the backward pass decomposes one global loss into per-parameter partial derivatives, so a single error becomes many local updates. Text space has no gradient, and the holistic alternative — "this rule seems wrong; rewrite it" — inherits both the measured attribution weakness of § 2.6 and an unbounded judgment cost per update. The entry contract is what restores tractability: **the pentad is the coordinate system in which text-space credit assignment becomes local.** Each signal class carries a characteristic *slot signature* — the accumulated evidence does not merely say an entry is unhealthy; it says which slot is indicted, and usually in which direction:

Accumulated evidence	Indicted slot	Slot-local operator
Routed-but-not-applicable dominates (applied ÷ considered low)	Activation — precision	Narrow the hook; grow the negative edge
Fired-off-map — applied outside the declared domain	Activation — boundary	Widen or re-shape the hook; the declaration was wrong, not the firing
Should-have-fired, surfaced by a defect backtrace or a recall probe	Activation — recall	Re-key; add a routing family; audit for the structural zero
Recalled, applied — and the steer still corrected the outcome	Payload	Re-abstract (altitude fault) or re-derive (content fault)
Clean application on one recurring sub-shape, never on another	Payload — fused	Split: leaf the entry (below)
A falsification-list fact reversed; a revisit trigger fired; anchors rotted	Warrant	Re-adjudicate; upgrade or drop; status flip
A defect shipped green through the gate	Enforcement	Grow the floor if the failed part is newly mechanical; otherwise re-disclose the residue honestly
The gate fires on correct behavior	Enforcement — over-reach	Shrink the floor to the invariant it serves
Never fired across review periods; domain no longer entered; duty fully absorbed by a floor	Lifecycle	Demote, retire on mootness, or de-enroll by coverage migration — after the killer-item check (I7)

Three properties give this procedure its backward-pass character while keeping it honest. **Locality:** updates arrive pre-localized. The pass never asks "is this entry good?" — it asks

"which slot does this evidence indict, and which way?" — and that is precisely what makes the owner-as-optimizer asymmetry (§ 3) feasible at human frequency: each judgment adjudicates a slot-local proposal with its evidence attached, never a rewrite. **Typed sparsity:** unlike a gradient, the signal is sparse and typed, and the noise filter runs first, so no slot updates on an irreducible failure. The division of labor inside an update is exact: the evidence *type* supplies the direction, the recurrence *count* supplies the nomination, and the human supplies the magnitude and the verdict. **Floors, not rates:** several rows read detection-limited streams (observed ≠ occurrence), and one — should-have-fired — has no complete instrument at all (§ 7, item 5); the pass treats those terms as lower bounds. The backward pass is honest about the partials it cannot estimate.

Split and fold: leafing

Two operators act on entry *granularity* rather than on a single slot, and the same telemetry nominates them. When an entry's evidence is **heterogeneous by sub-shape** — dispositions bimodal across recognizable sub-cases, applying cleanly on one and never on the other — the entry is fused, and the operator is a **split**: leaf it into children, each of which must re-satisfy the whole pentad (its own hook, duty, warrant, floor, and lifecycle), with the parent either retiring by coverage migration or surviving as the leaves' shared warrant. The reverse signal is **covariance**: two entries whose answers converge across items put their differentiation under stress — covariance is the watch signal; entailment (one entry's answer settling the other's with no further judgment) confirms the collision — and the operator is a **fold**. The default bias is fold-before-split, because a premature split mints two lifecycles where one suffices; and every split or fold is ratified against the raw anchors, never against the labels — § 5.6's staged-minting discipline applied to entry boundaries.

The latch tier under the same analysis

The scheduled checkpoints of § 5.5 are pentad-shaped, so their telemetry factors identically. Fired/not-applicable distributions and off-map records indict the hook: re-scope it. Answer entailment between two latches' duties is the fold signal: merge, because exactly one latch owns any judgment. A latch whose judgment residue has emptied — its paired gate now owns everything it checked — de-latches to a pure gate, probationary. A latch whose consulted authority has dissolved retires with it. One reading rule guards the tier: distributions are **signature-relative** — a checkpoint firing hard in a period whose work is its class is the system working, not an anomaly to prune. And the feedback closes in both directions, which is § 5.5's complementarity made operational: per-latch dispositions flow into the ledgers, and ledger consolidation passes are the only legitimate minters of new hooks.

Every threshold, at every tier, pairs with this scheduled reverse (I7): applied ÷ considered as nominator, never verdict; mootness and coverage migration as the retirement keys, never low salience. A system that runs the up-pipeline without scheduling the down-pipeline accumulates confidently-wrong rules — CBR's utility problem (Smyth & Keane, 1995) and Wikipedia's ossification (Halfaker et al., 2013) arriving at the same lesson from two directions.

5.5 The read side: latches

The ledger pipeline produces distilled knowledge; **latches** schedule its re-application. A latch is a cued-recognition checkpoint at plan-assembly time — itself pentad-shaped: ⟨a fires-when predicate over work-shape · the one authority consulted · a bounded judgment duty · a typed record with a named consumer · a gate checking the mechanical remainder⟩. The latch walk is the session's *query*: the fixed join schedule that assembles working context from the stores. Boundary tests keep the tier honest: a latch with no judgment residue is a gate; a recallable constraint is a rule; a latch exists only for a distinct authority join or a generative procedure a rule sentence cannot carry. Hooks may overlap — work-shapes co-occur, and forced partition manufactures misses — but duties must be disjoint: exactly one latch owns any judgment.

Latches and ledgers are complements, and the complementarity is the design. Latches without ledgers: the system recalls its distilled judgment on schedule and never learns where that judgment is wrong — a confident recall cache, rotting invisibly. Ledgers without latches: adjudicated lessons with no scheduled moment of re-application — a well-audited graveyard. Joined: consolidation passes over the ledgers are the only legitimate minters of new latch hooks, rules, and gates; the latch walk is where minted knowledge earns its keep; and per-latch dispositions feed back into the ledgers. The loop closes.

5.6 Escape hatches: how closed sets stay honest

Every closed vocabulary in the system — verdict sets, class registries, routing families — ships an honest escape (`other(<what>)`, `unresolved(<carry>)`). This is not tolerated slack; it is the vocabulary's own measurement channel. A forced pick manufactures noise in both directions at once — inflating a wrong class's recurrence count while keeping the missing class invisible — and since recurrence drives every structural update, a vocabulary without an escape corrupts the exact quantity the loop consolidates on. Two same-shaped escapes mint the missing class: the vocabulary grows from its own residue stream, evidence-first. And the anti-pattern that wears rigor's costume is named: a closed set whose only "escape" is loud rejection at ingress. Rejection is a gate; an escape is an instrument; a

healthy closed set has both. Escapes are per-level in any routing hierarchy, because a set's need to grow is knowable only from its own escape stream.

The escape law sits inside a fuller birth procedure, because class sets are stage-one keys (§ 4.3) and their birth method matters. Two kinds of phenomenon space, two births. Where a closed decision procedure encloses the space, the class set is **derived** — the leaves of the decision tree, total by construction — and a recurring escape there does something stronger than propose a label: it indicts the procedure that claimed to enclose the space. Where no procedure encloses the space (kinds of mistakes, kinds of steers, kinds of work), minting is **staged**: free prose until instances exist, because classes invented ahead of instances are priming, not observation; then organic labels with mandatory anchors — the label a hypothesis, the anchored instance the datum, so relabeling stays cheap; then blind-coded ratification at two to three recurrences (§ 5.3's coding species); then a closed registry with loud-unknown ingress, whose escape stream is thereafter its only legitimate growth channel.

5.7 The carrier: the fast loop

The machinery above is cross-session — the slow loop. What carries it is the **fast loop**: the session, the only place where knowledge is actually exercised and where every signal the slow loop consumes is generated. Its anatomy is the circuit that joins the statics of § 4 to the dynamics above, and a project that adopts the stores without it has built a library, not a learning system.

A session opens by **assembling its context**, and the assembly is the read side of everything § 4 built, run as one procedure: the constitutional tier loads unconditionally; the trace store's carry-forward header restores cross-session continuity; a brief orientation classifies the work's shape; and the latch walk runs the fixed join schedule (§ 5.5) — routed rules pulled by their declared firing shapes, case summaries scanned with revisit triggers checked, section-mapped documents partially read. The assembled window is the query's result set, under § 4.3's cost discipline.

Work then proceeds with micro-decisions delegated to the strongest available adjudicators — compiler, tests, the live tree (§ 6.2) — and verification is behavioral (§ 3, doctrine 5). As work runs, candidate signal is jotted at the episodic floor: one noticing, one anchor, seconds of cost (§ 5.4's cheapest bar). Capture must stay near-free, or the capture problem (Lee, 1997) wins.

The session **closes by producing the slow loop's inputs** — the step most pile-based practice omits. A reflect pass leads with the session's typed event streams before free recall; the query's returns get per-entry **use-time dispositions** — applied, considered-but-not-applica-

ble, fired-off-map — the telemetry the demotion review reads (§ 5.4); candidate observations and caught contradictions file to their owning ledgers; and a session log appends to the trace store. The reflect step terminates at the human — the steer channel's scheduled moment (§ 5.1) — and the close is where the separation of powers cashes out daily: the session proposes, and nothing it proposes is yet true.

Boot consumes what the slow loop produced; close produces what the slow loop will consume. That closure — not any individual store — is the system.

5.8 The pipeline, worked end-to-end

The refinement law in motion: one generic lesson traced through every tier. Watch who acts at each step — agents notice, draft, and nominate; the world contradicts; every movement between tiers is the human's (§ 3).

Noticing (bar: one instance). A session, mid-task, notices that a retry wrapper discards the underlying error's cause when it rethrows. Not worth stopping for; not nothing. One line lands in the episodic pile — *observed: wrapped errors in the ingest path lose their cause; re-check when touching error wrapping* — with a file anchor. No classification, no decision, no process change: the entry bar sits deliberately at the floor, because everything above it filters.

Promotion (episodic → case). Weeks later a second session hits the same shape in the export path and appends its own line. The episodic pass — trigger-fired when a re-check condition matches, periodic otherwise — groups the pile by shape and finds two anchored instances of one undecided fork. Two instances is this move's bar, and the pass fixes nothing: it surfaces a **promotion candidate** — recurring and undecided, so someone must decide it once. A decision entry is drafted in the case store's shape (options considered; constraining facts as a falsification list), faces the write-time examiner (the attack species, § 5.3), and then the human performs the promotion. The promotion **raises abstraction** (I6): the decision says *errors that wrap must carry their cause* — object-decoupled — not *fix ingest and export*; tier-copying the instances would build a recall cache, not a policy. The two observations are deleted-with-pointer; the pile empties *by* promotion.

Proceduralization (case → rule). The decision binds only if recalled, and its domain is entered constantly — every catch block that rethrows. The tell that a case needs a rule form: a session is observed **not propagating it** — the decision was settled, discoverable, and still missed, because full-scan recall does not fire reliably inside implementation flow. That observed non-propagation nominates; the human's approval promotes. The rule is the decision's duty form — *fires-when: authoring a catch that wraps or rethrows*, the duty sentence

carrying the obligation, the decision cited as warrant. The case keeps the *why*; the rule carries the *when-and-what*; nothing is duplicated that could drift. This middle operator — case → production rule — is the load-bearing one in the whole pipeline: it converts a thing the project *knows* into a thing that *fires*.

Mechanization (rule → floor). Later, part of the duty proves fully mechanical — *the wrapping constructor receives a cause argument* is lintable. When that mechanical part has failed silently twice (the silent-failure bar), a gate takes it. The rule's floor grows; its residue shrinks and is re-disclosed: the gate now checks that *a* cause is passed, and judgment still owns whether it is the *right* one (I10).

And back down, on the calendar. At the periodic review, each rule's dispositions (§ 5.7) are read — applied ÷ considered as nominator, never verdict. A rule routed constantly and rarely applicable gets its hook re-scoped. A rule whose gate fully owns its duty de-enrolls by coverage migration — it lives on at its enforcement surface, the decision staying behind as warrant. The case itself retires only on mootness. Every one of these moves is the human's; the counters only nominate.

The trace also shows structurally what § 6 argues in general: at each handoff, pentad slots change hands exactly as the coordinates predict — the noticing had only activation and anchor; the decision added a warrant; the rule added routable activation and an enforcement obligation; the gate is all floor. The families are stations on one line.

6. One contract, many families: GCE across artifacts and agents

The user-facing diversity of a mature project's knowledge artifacts — rules, decision records, defect ledgers, checklists, gates, role definitions, episodes — is, under GCE, one contract evaluated at different coordinates (§ 4.5). This section works the claim family by family; it is the paper's demonstration that the doctrine is a *theory of the artifact space*, not a collection of practices.

6.1 The coordinate table

Pentad slot	Rule entry	Decision entry (ADR)	Latch	Ledger record	Gate	Observation
Activation	routed: declared firing shapes + a negative edge	one-line summary, full-scanned; revisit triggers as re-activation	hook walked on schedule at plan time	the contradiction's arrival (species-dependent)	unconditional over a derived population	trigger-fired ("re-check when…")

Pentad slot	Rule entry	Decision entry (ADR)	Latch	Ledger record	Gate	Observation
Payload	condition → duty, one sentence, object-decoupled	the Decision — a constraint on future forks	a bounded judgment procedure	claim-as-held + verdict	all floor	a noticing
Warrant	enrollment evidence + human approval	constraining facts written as a falsification list	the one authority × its truth-maker	mandatory anchors + the unstaked adjudicator	derivation of its checked population	the anchor
Enforcement	explicit floor + disclosed residue	shape-gated only; content deliberately judgment	the latch's paired record gate	record grammar mechanical; verdict judgment	total (residue zero by definition)	none
Lifecycle	applied ÷ considered → demotion review; tombstones	status flip / supersedure; retirement on mootness	disposition lines + retirement by grounding class	classification pass + attacker-precision review	class floors; retirement on supersession	delete-with-pointer on promotion

Reading the table columnwise recovers each family's felt character; reading it rowwise shows the same slot flexing under different coordinates. Two worked contrasts make the mechanism concrete.

Why a rule is not a decision. Rule recall is associative over dozens of heavy duties, so activation must be a *routable key* — and any router over-fires, so the slot grows a precision control (the negative edge: "not-this"). The payload is a duty that must *bind at work time*, so the enforcement slot is obligatory and explicit — the floor the gate checks, and the residue the reader must still carry. Decision recall is a bare full scan of cheap summaries, so activation *is* the summary line and needs no negative edge; the payload constrains future design rather than commanding present work, so enforcement nearly vanishes (shape gates only) and the *warrant* slot dominates instead: the constraining-facts section is written as a falsification list because "is this still correct?" is the case store's live question, where "did you actually do it?" is the rule store's. Same contract; the recall economics and the live question set the coordinates. What this upgrades in the industry's folk forms is precisely the slots the folk forms silently lack: ADRs nobody can tell are stale (no currency channel), rules nobody recalls at the right moment (no activation discipline) — the completeness obligation is the contribution, not the artifact.

Why ledger records are deliberately key-poor. The two-tier, progressively-disclosed shape that serves rules and decisions is *store-relative*, not universal, and the ledger piles prove it. Ledger records get minimal frontmatter and no routing tier, for three ascending reasons: no mid-work retrieval exists to serve (work-time reads consume the ledger's con-

solidated successor — the ratified class, the minted rule — never the raw record); the consolidation read is wholesale by design (recurrence counting and observed≠occurrence floors are population properties, and a per-record routing surface optimizes a read that must never be partial); and *blindness preservation* — the consolidation pass runs a blind second coder over raw claims and anchors *without* the fast-loop's candidate labels (§ 5.3, coding species), and a retrieval-tuned key tier would surface exactly those labels to exactly the reader who must not see them. The general law: ask of every store, *who reads this, when, how many entries at once — and must any reader be blind?* Individually, on demand, into a bounded context → two-tier, disclosed, routed. Wholesale, at a scheduled pass → flat, append-cheap, key-poor on purpose.

The projection seam. Every entry splits into a machine surface (flat metadata) and a judgment surface (the body); indexes, queues, and status tables are read models regenerated from machine surfaces, never hand-maintained. The slot mapping is exact: **projections carry activation and lifecycle — never payload or warrant.** A projection cell may invite a read; it must never settle a question. Consuming a conclusion from a summary cell or status table is a named failure — *authority leak*: decided-nowhere-but-recorded-somewhere, a conclusion acquiring decision authority through citation of a surface that carries no warrant.

6.2 Agent families: roles as independence machinery

Sub-agent roles exist in GCE for an epistemic reason before any throughput reason: **independence is structural, not attitudinal.** A single context cannot be instructed into genuine self-skepticism — the measured asymmetry runs against the author (§ 2.1) — but skepticism can be *constructed* by giving the contradiction its own context with no authoring stake. The role taxonomy therefore follows the ledger taxonomy, not an org chart:

- **Examiners** are contradiction generators (attack species): dispatched against proposed plans and proposed store entries, refute-phrased, reporting proposed records with verdicts left pending. They never write or commit; proposals flow to the adjudicating session and the human gate.
- **Independent derivers** are collision parties: canonically, a test author with no authoring stake derives expectations blind-first from committed artifacts, diffs its derivation against the implementation's claims, reports every discrepancy, and quarantines rather than adjudicates its own disagreements.
- **Reconnaissance** grounds claims against the live system — the world channel, feeding verified facts into every other party's records.
- **The implementer deliberately keeps no ledger** — and the omission is the pattern working. An implementer holds no independent belief worth ledgering: its micro-deci-

sions are intentionally delegated to the strongest available adjudicators — the compiler, the tests, the tree — and a delegated micro-decision re-derived against reality is the design operating. When the implementer catches its *briefing* wrong, the finding files in the plan's ledger: one owning ledger per domain — a finding lives where the failed belief lives, not where the finder sits.

What makes the implementer's ledgerlessness safe is a partition drawn *per fork* at plan time, not a global division of labor. A pending fork is plan-side iff it is a one-way door (irreversible or expensive to reverse; Bezos, 2016), its coupling fans out past one owner, or committed artifacts already answer it. It is implementation-side iff the tree, the compiler, or the tests will adjudicate it better than deliberation would — the delegate-to-the-strongest-adjudicator principle above, kin to deciding at the last responsible moment (Poppendieck & Poppendieck, 2003). Forks that fit neither test — plan-time-*invisible* — are the hazard, and they receive a third, typed disposition instead of a silent gamble: an **information purchase** (a probe buys the fact before the fork is forced) or a **declared-open** carry with a STOP condition the implementer *surfaces and never decides*. The partition is empirically estimable from the defect streams (§ 5.3, reality species): over-scoped plans show up as plan-asserted micro-decisions falling to the tree; under-scoped forks show up as STOP events or — decided silently and wrongly — as defects with zero prior paper trail, the discriminator that separates silent planning errors from planned evolution.

Two disciplines keep the roles honest at scale. **Freeze the target**: an attack runs against the verbatim text it was handed, never a paraphrase; a colliding derivation runs blind before reading the other party's claims. **Derive, never copy**: a specialized role definition reads its base definition live and overrides named sections rather than carrying a copy — the single-canonical-source law (§ 4.4) applied to the roles themselves. And one honesty discipline binds every role's output: **a written deferral is a success outcome** — a property the current instrument cannot check is declared, with what would verify it and when, never faked with a weaker check that happens to pass.

The blackboard architectures of the 1980s put independent knowledge sources in front of a shared hypothesis space with an arbiter (Erman et al., 1980); GCE's role system is that idea with two corrections the intervening decades taught: the arbiter must be unstaked, and the hypothesis space must persist — and be governed — across episodes.

6.3 The episode family

Multi-session work runs in **episodes**: premise-locked units with a living plan-spine, just-in-time slice materialization, and a convergence ceremony. The episode layer is where the ledger philosophy meets planning:

- **Premise-lock + finite convergence.** Goals are the convergence yardstick, verified against reality at close — never task-list fulfillment. Converged episodes never re-open; a changed premise claims a *new* episode seeded with the old one's explored ground. The lock is what makes convergence falsifiable — a mutable destination converges on whatever happened — and, as a side effect, what makes unplanned change detectable: a surface changing under a live episode is planned work; the same change with no episode is drift. The human-practice cousin is Shape Up's circuit breaker (Singer, 2019), not any AI tool: the surveyed spec-driven products run living specs that re-open by design (§ 2.3), and the divergence is deliberate.
- **Plan on-policy.** The spine holds intent; concrete work items materialize only for the next one or two slices, after prior slices' reality is observed. Stored plans rot; minimize stored-plan inventory and re-derive the next step from the current state under fixed goals.
- **The spine/log split.** The spine is a rewrite-in-place statement of the *what* — settled design under stable headings, no correction lineage; the logs are the append-only *how*. This is the event-sourcing segregation (§ 4.4) applied to plans, and it is what keeps working memory from degrading into a second episodic pile.
- **Boundary consolidation.** Ceremonies at close — surprise scoring against the frozen predictions (the forecast ledger, § 5.3), pain-versus-hypothesis review, fold-back of distilled shapes into the cross-episode catalog — because boundaries are where the human is present, and the reflect step terminates at the human. This is a third consolidation trigger family beside the field's idle-triggered (sleep-time compute, 2025) and accumulation-triggered (Park et al.,

2023. designs, chosen precisely for its governance property.

6.4 The instantiation law

Summarizing the section as doctrine: **a new artifact class or agent role is admitted by filling the pentad and locating its coordinates — and by finding, for each slot, which existing family it borrows discipline from.** A proposed artifact that cannot say when it activates has designed a structural zero; one whose warrant cites itself has designed a self-warranting verifier; one with no consumer has designed a graveyard; a role that adjudicates its own proposals has collapsed the separation of powers. The families are not a fixed roster but a coordinate space with its failure modes mapped; growth means claiming a new point in the space, under the same contract.

6.5 Genesis: the roster is an output

One question remains for the section's claim to be complete: where does the coordinate space's *population* come from? The doctrine's answer: **the organ roster is an output of the loop, not an input.**

A project adopts GCE by seeding the smallest configuration that can run one full cycle: the invariant set and the entry contract (§ 4.5); the separation of powers (§ 5.2); a constitutional file with its hard cap; one episodic pile with mandatory anchors; a case store; a trace. Everything else — which rules, which latches, which gates, which ledger species beyond the reality ledger every project already has whether it admits it or not — is minted later, by the loop, from the project's own typed miss stream, through the route-before-mint ladder, at the project's own bars.

Two ordering rules govern the bootstrap. **Start the ledgers before the latches**: a young project has few distilled lessons to schedule but generates contradictions from day one, and the ledger discipline is what eventually reveals which checkpoints, rules, and gates *this* project needs — the latch roster is late-bound by design. And **never copy another project's roster**: a roster encodes its origin's failure history, so importing one imports a standing tax calibrated to someone else's misses — which is why both § 2.7's precedents and the origin system's own roster are offered throughout as evidence, never as templates. The origin system's genesis followed this shape observably: its checkpoints were born event-anchored — a failure, a retrospective finding, a cluster of corrections — and arrived in codification waves when accumulated records revealed a systematization, not on a growth-proportional cadence.

The bootstrap also discharges an apparent circularity in § 6.4: the instantiation law presupposes a running loop, and the loop's first organs cannot have been admitted by it. They need not be — they are the seed, stated above, and every organ after the seed is loop-minted. The system is self-hosting in the compiler's sense: hand-built once, at minimum viable size, then grown only through its own governed pipeline.

7. Failure surface and boundary conditions

Stated honestly, because the advantages and the costs are one design viewed from two sides.

1. **Hand-maintained where products mechanize.** Much of what GCE runs as conventions-plus-gates, tooling products ship as features. Real, permanent cost, partially amortized by projection machinery — and the design-rationale literature's capture problem

(Lee, 1997) is the standing threat: the discipline survives only while capture stays near-free, which is why agent-drafted/human-gated is the only viable labor split.

2. **Precondition-bound.** The approach pays off under one consistent calibrating judge, many episodes, and a long horizon. Outside those preconditions the discipline is tax. Solo calibration is simultaneously the enabling precondition and the largest validity threat: n=1 judge, no inter-rater check.
3. **Presence, not quality.** The mechanical floor checks that records exist, parse, and carry their slots; prediction quality, completeness judgment, and vocabulary goodness stay unmechanized *by design* — which means the system's core quality claims are exactly its unmeasured ones.
4. **The overhead tax.** Bookkeeping is time off product work and is the dominant felt cost. The correct response is directional: mechanize bookkeeping relentlessly; never mechanize the completeness judgment.
5. **Silent misses are floors.** Every detector's observed-zero is detection-limited; the write side of the slow loop has no complete miss census. Recall — an entry that should have fired but was neither routed nor recalled — is unmeasurable without ground truth; the only honest answers are independent-judge probes over labeled sets and the passive archaeology of downstream defect stores.
6. **Forgetting is relocated, not eliminated.** Weight-forgetting is traded for retrieval failure; structural zeros are the new forgetting (the continual-learning-to-memory literature documents the same relocation in auto-written stores). And the frozen model is a dependency that version-bumps: a model swap silently re-prices every checkpoint and every rule sentence's interpretability, in both directions — duties can become unnecessary, and surfaces that needed none can come to need one. Price the swap deliberately; watch referents, not aliases — a drift watcher over an indirection layer measures the pointer's stability and reports it as the referent's.
7. **Recall, not transfer — by design and by boundary.** The gains are in-domain — external-memory gains measured elsewhere are likewise recall within problem families rather than cross-distribution transfer (Beyond Perplexity, 2026) — and the abstraction-on-promotion step is the one intervention pushing past pure recall. Cross-project transfer of the *system* — as opposed to its knowledge — is exactly what an independent adoption would test, and none has yet.
8. **Threshold currency under-derived.** The two-to-three-instance bars are house convention; the field's alternatives (vote counts, importance sums, helpful/harmful counters) differ and derive their currencies no better. Per-memory utility attribution is an active

research direction (Yuan, Su & Yao, 2026; and successors), and a richer currency — applied ÷ considered ratios — is the nearest in-house instrument.

8. Discussion

8.1 What GCE is not

Not self-evolving context — it is that field with the write path constitutionally removed from the model (§ 2.1). **Not RAG** — the read path is deliberately RAG-shaped; the write-side learning lifecycle is what RAG lacks, and the retrieval keys are symbolic on purpose (§ 2.4). **Not fine-tuning by other means** — the equivalence fails at mechanism; what survives is the payload-slot claim that favors context anyway (§ 2.5). **Not automated prompt optimization** — GCE refuses the closed text-update loop at the attribution step and keeps the human as the optimizer (§ 2.6). **Not spec-driven development** — the surveyed tools run living specs, upfront task generation, and mechanized convergence; GCE runs premise-locked finite episodes, just-in-time materialization, and convergence held as human judgment (§ 6.3). **Not a project-management methodology and not a product** — it borrows named pieces of human practice as vocabulary, never as imported apparatus, and much of what products ship as features it runs as conventions plus gates, a cost § 7 names first. **Not universal** — the preconditions are stated, and outside them the discipline is tax.

8.2 The convergence argument

The paper's strongest external evidence is not any single citation but a pattern across them. The origin system's countermeasures — append-only piles, typed stores, delta-not-rewrite, hard-capped always-loaded tiers, human-gated writes — were in place before the literature measured the failures they prevent: context collapse (2025), heterogeneous-store contamination (2026), longitudinal memory risk (2026), context rot (2025), memory poisoning (2024–2025). Independently, the tool industry converged on GCE's loading taxonomy without its governance layer (§ 2.3), and the memory field's own 2026 turn — toward typing, provenance, per-memory utility, and quality governance — is a turn *toward* the positions argued here. Convergent evolution from opposite directions is the strongest validation available to a doctrine that cannot run controlled trials on itself. It is not proof; § 7's limits stand.

8.3 On borrowed vocabulary

The system was first understood through ML-training analogies, and each borrowed term was graded against the literature before use; the record of rejections is part of the doctrine.

"The documents are fine-tuning" fails at mechanism (§ 2.5). "Verification is backpropagation" fails at attribution (§ 2.6); the analogy's one surviving form maps governance polarity — sessions as the forward pass, the world as the loss signal that contradicts but never writes, agents as proposed updates, the human as the optimizer step — never update mechanism. The discipline generalizes: analogies supply vocabulary, pathology taxonomies, and specification language; they never justify apparatus. A doctrine that names its own metaphors' breaking points is harder to mis-apply — which is why this subsection exists.

8.4 The completeness posture

A doctrine that presents itself as a system owes an account of whether its enumerations are complete — six stores, six observed contradiction species, five slots, sixteen invariants. The honest answer is that GCE makes two different completeness claims and marks which is which. Where a closed derivation encloses a space, totality is by construction: the pentad's five slots are the five global functions the removed optimizer no longer provides, and a sixth would have to arrive as a sixth function — none has (§ 4.5). Everywhere else — the store roster, the species table, the artifact families, the invariant set itself — the enumeration is **closed-with-escape** (§ 5.6): complete relative to the observed miss stream, with the instrument that detects incompleteness built in. A seventh contradiction species claimed, a recurring escape in any registry, an organ born needing a missing slot — each is a standing trigger that indicts and re-derives its taxonomy, and the instruments themselves are instrumented (I16). "Rationally complete as far as we know" is therefore a precise, falsifiable posture rather than a hedge: the doctrine's own taxonomies obey the doctrine, and their incompleteness, when it arrives, is designed to arrive as a countable event rather than a surprise.

8.5 Open problems

Four are honest research questions rather than engineering backlog. **Recall measurement**: should-have-fired ground truth for routed knowledge, without exhaustive labeling — the passive defect-archaeology estimator proposed in § 7.5 has run in one system, once. **Threshold derivation**: whether promotion bars can be derived from measured attribution noise and per-entry utility rather than set by convention. **Vocabulary timing**: taxonomies serve communication (where late codification is safe) and perception (where a judge without the category cannot see the category error); no located guidance resolves when a class set must precede the judging it serves. **The transfer test**: the doctrine's generality claim is falsifiable exactly one way — a second, independent adoption growing its own organ roster from its own miss stream under the same invariants — and that experiment is, by construction, not ours to run.

9. Conclusion

Under the frozen-model constraint, the question "how should an AI-assisted project manage its documentation?" is mis-posed. The corpus is not documentation *about* the system; it is the only substrate in which the system can learn. Taking that seriously yields the doctrine argued here: unbundle what training bundles; give every store its own decay profile and every entry the full five-slot loop that nothing global will run for it; let contradiction come from structurally independent sources and be judged by unstaked parties; damp refinement with recurrence bars graded by blast radius; and keep a human at the write gate — not as a concession to caution but as the one write policy the surveyed landscape has not falsified. The model's intelligence is rented. The learning had better be owned.

References

Preprints are cited by arXiv identifier; practice sources by primary URL. Where author lists were not verifiable from primary sources, entries are given by title.

ACE — *Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models.* arXiv:2510.04618, 2025.

adr.github.io — *Architectural Decision Records.* GitHub ADR organization. https://adr.github.io/

AGENTS.md — *AGENTS.md: a simple, open format for guiding coding agents.* OpenAI et al., 2025; Agentic AI Foundation / Linux Foundation. https://agents.md/

AgentPoison — *AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases.* NeurIPS 2024.

Agrawal, L. et al. — *GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning.* arXiv:2507.19457, 2025.

Aamodt, A. & Plaza, E. — *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* AI Communications 7(1), 1994.

Alchourrón, C., Gärdenfors, P. & Makinson, D. — *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions.* Journal of Symbolic Logic 50(2), 1985.

Al-Tawaha et al. — *Remembering More, Risking More: Memory-Induced Risk in LLM Agents.* arXiv:2605.17830, 2026.

Anderson, J.R. & Schooler, L.J. — *Reflections of the Environment in Memory.* Psychological Science 2(6), 1991.

Anthropic — *Effective context engineering for AI agents.* Engineering blog, September 2025. (2025a)

Anthropic — *Agent Skills.* October 2025; cross-vendor standard December 2025. (2025b)

Anthropic — *Claude Code: Manage Claude's memory.* https://code.claude.com/docs/en/memory

ArcMemo — *Abstract Reasoning Composition with Lifelong LLM Memory.* arXiv:2509.04439, 2025.

Argyris, C. — *Double Loop Learning in Organizations.* Harvard Business Review 55(5), 1977.

Beyer, B., Jones, C., Petoff, J. & Murphy, N.R. (eds.) — *Site Reliability Engineering.* O'Reilly, 2016. Ch. 15, "Postmortem Culture."

Beyond Perplexity — on external-memory gains as within-family recall. arXiv:2607.00368, 2026.

Bezos, J. — *2015 Letter to Shareholders* (Type 1 / Type 2 decisions). Amazon, 2016.

Bradner, S. — *The Internet Standards Process — Revision 3.* RFC 2026, IETF, 1996.

Butler, B., Joyce, E. & Pike, J. — *Don't Look Now, But We've Created a Bureaucracy: The Nature and Roles of Policies and Rules in Wikipedia.* CHI 2008.

Cartridges — *Lightweight, general-purpose long-context representations via self-study.* arXiv:2506.06266, 2025.

Chan, S. et al. — *Data Distributional Properties Drive Emergent In-Context Learning in Transformers.* arXiv:2205.05055, 2022.

Chhikara, P. et al. — *Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.* arXiv:2504.19413, 2025.

Conklin, J. & Begeman, M.L. — *gIBIS: A Hypertext Tool for Exploratory Policy Discussion.* ACM TOIS 6(4), 1988.

Dai, D. et al. — *Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers.* arXiv:2212.10559, 2022.

de Kleer, J. — *An Assumption-based TMS.* Artificial Intelligence 28(2), 1986.

Doyle, J. — *A Truth Maintenance System.* Artificial Intelligence 12(3), 1979.

Erman, L.D., Hayes-Roth, F., Lesser, V.R. & Reddy, D.R. — *The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty.* ACM Computing Surveys 12(2), 1980.

Fang, Peng et al. — *A Comprehensive Survey of Self-Evolving AI Agents.* arXiv:2508.07407, 2025.

Feigenbaum, E.A. — *The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering.* IJCAI-77, 1977.

Forte, A. & Bruckman, A. — *Scaling Consensus: Increasing Decentralization in Wikipedia Governance.* HICSS-41, 2008.

Gao, H. et al. — *A Survey of Self-Evolving Agents.* arXiv:2507.21046, 2025.

Gawande, A. — *The Checklist Manifesto.* Metropolitan Books, 2009.

GitHub — *Spec Kit: spec-driven development toolkit.* https://github.com/github/spec-kit, 2025.

Ha et al. — *MemGuard: typed functional roles against heterogeneous memory contamination.* arXiv:2605.28009, 2026.

Halfaker, A., Geiger, R.S., Morgan, J. & Riedl, J. — *The Rise and Decline of an Open Collaboration System.* American Behavioral Scientist 57(5), 2013.

Haynes, A.B. et al. — *A Surgical Safety Checklist to Reduce Morbidity and Mortality in a Global Population.* NEJM 360(5), 2009.

HippoRAG — *Neurobiologically Inspired Long-Term Memory for Large Language Models.* arXiv:2405.14831, 2024.

Hong, K., Troynikov, A. & Huber, J. — *Context Rot: How Increasing Input Tokens Impacts LLM Performance.* Chroma technical report, July 2025.

Karpathy, A. — *On system prompt learning.* X, May 10, 2025.

Kiro (AWS) — *Steering.* https://kiro.dev/docs/steering/, 2025.

Laird, J.E., Rosenbloom, P.S. & Newell, A. — *Chunking in Soar: The Anatomy of a General Learning Mechanism.* Machine Learning 1(1), 1986.

Leake, D.B. & Wilson, D.C. — *Categorizing Case-Base Maintenance: Dimensions and Directions.* EWCBR 1998, LNCS 1488.

Lee, J. — *Design Rationale Systems: Understanding the Issues.* IEEE Expert 12(3), 1997.

Lenat, D.B. — *CYC: A Large-Scale Investment in Knowledge Infrastructure.* CACM 38(11), 1995.

Letta — *Sleep-time compute.* letta.com, 2025; see also *Sleep-time Compute: Beyond Inference Scaling at Test-time.* arXiv:2504.13171, 2025.

Li, Z. et al. — *MemOS: A Memory OS for AI Systems.* arXiv:2507.03724, 2025.

Luo, J. et al. — *From Storage to Experience: A Survey on the Evolution of LLM Agent Memory Mechanisms.* arXiv:2605.06716, Findings of ACL 2026.

Lütke, T. — on context engineering. X, June 19, 2025; see also Willison, S., *Context engineering.* simonwillison.net, June 27, 2025. (Willison, 2025a)

Mei, L. et al. — *A Survey of Context Engineering for Large Language Models.* arXiv:2507.13334, 2025.

memorywire — *A vendor-neutral wire format for agent memory operations with optional human-in-the-loop governance.* arXiv:2606.01138, 2026.

MINJA — *Memory Injection Attacks on LLM Agents.* arXiv:2503.03704, 2025.

Nonaka, I. — *A Dynamic Theory of Organizational Knowledge Creation.* Organization Science 5(1), 1994.

Nosek, B.A., Ebersole, C.R., DeHaven, A.C. & Mellor, D.T. — *The Preregistration Revolution.* PNAS 115(11), 2018.

Nygard, M. — *Documenting Architecture Decisions.* November 2011.

Ovadia, O. et al. — *Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs.* arXiv:2312.05934; EMNLP 2024.

OWASP — *Agentic AI Top 10: ASI06, Memory and Context Poisoning.* 2026.

Packer, C. et al. — *MemGPT: Towards LLMs as Operating Systems.* arXiv:2310.08560, 2023.

Park, J.S. et al. — *Generative Agents: Interactive Simulacra of Human Behavior.* arXiv:2304.03442, 2023.

Poppendieck, M. & Poppendieck, T. — *Lean Software Development: An Agile Toolkit.* Addison-Wesley, 2003.

Procida, D. — *Diátaxis.* https://diataxis.fr/, 2020.

PROJECTMEM — *A local-first, event-sourced memory and judgment layer for AI coding agents.* arXiv:2606.12329, 2026.

Pryzant, R. et al. — *Automatic Prompt Optimization with "Gradient Descent" and Beam Search (ProTeGi).* EMNLP 2023.

Rasmussen, P. et al. — *Zep: A Temporal Knowledge Graph Architecture for Agent Memory.* arXiv:2501.13956, 2025.

Semantic backpropagation — *Semantic Backpropagation for Language-Based Agentic Systems.* arXiv:2412.03624, 2024.

Shen, L. et al. — *Do Pretrained Transformers Learn In-Context by Gradient Descent?* ICML 2024; arXiv:2310.08540.

Shinn, N. et al. — *Reflexion: Language Agents with Verbal Reinforcement Learning.* arXiv:2303.11366, 2023.

Singer, R. — *Shape Up: Stop Running in Circles and Ship Work that Matters.* Basecamp, 2019. https://basecamp.com/shapeup

Singh, A. et al. — *The Transient Nature of Emergent In-Context Learning in Transformers.* arXiv:2311.08360, 2023.

Smyth, B. & Keane, M. — *Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems.* IJCAI 1995.

Sumers, T. et al. — *Cognitive Architectures for Language Agents (CoALA).* arXiv:2309.02427, 2023.

Suzgun, M. et al. — *Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory.* arXiv:2504.07952, 2025.

ThoughtWorks — *Lightweight Architecture Decision Records.* Technology Radar, Adopt ring, November 2017.

Training-Free GRPO — *Training-Free Group Relative Policy Optimization.* arXiv:2510.08191, 2025.

von Oswald, J. et al. — *Transformers Learn In-Context by Gradient Descent.* arXiv:2212.07677, 2022.

Walsh, J.P. & Ungson, G.R. — *Organizational Memory.* Academy of Management Review 16(1), 1991.

Wang, G. et al. — *Voyager: An Open-Ended Embodied Agent with Large Language Models.* arXiv:2305.16291, 2023.

Wang, Z.Z., Mao, J., Fried, D. & Neubig, G. — *Agent Workflow Memory.* arXiv:2409.07429, 2024.

Wegner, D.M. — *Transactive Memory: A Contemporary Analysis of the Group Mind.* In *Theories of Group Behavior*, Springer, 1987.

When Continual Learning Moves to Memory — on forgetting relocated into retrieval, memory dilution, and instance-versus-insight transfer. arXiv:2604.27003, 2026.

Willison, S. — *The lethal trifecta for AI agents.* simonwillison.net, June 16, 2025. (2025b)

Xu, W. et al. — *A-MEM: Agentic Memory for LLM Agents.* arXiv:2502.12110, 2025 (NeurIPS 2025).

Xu, W. et al. — *Pride and Prejudice: LLM Amplifies Self-Bias in Self-Refinement.* arXiv:2402.11436, 2024.

Yang, C. et al. — *Large Language Models as Optimizers (OPRO).* arXiv:2309.03409, 2023.

Yuan, Su & Yao — *Diagnosing Retrieval vs. Utilization Bottlenecks in LLM Agent Memory.* arXiv:2603.02473, 2026.

Yuksekgonul, M. et al. — *TextGrad: Automatic "Differentiation" via Text.* arXiv:2406.07496, 2024.

Zhang, S. et al. — *Which Agent Causes Task Failures and When? (Who&When).* arXiv:2505.00212, 2025.

Zhang, Z. et al. — *A Survey on the Memory Mechanism of Large Language Model based Agents.* arXiv:2404.13501, 2024; ACM TOIS 2025.

Zhao, A. et al. — *ExpeL: LLM Agents Are Experiential Learners.* AAAI 2024; arXiv:2308.10144.

Zhong, W. et al. — *MemoryBank: Enhancing Large Language Models with Long-Term Memory.* arXiv:2305.10250, 2023; AAAI 2024.

Provenance note. This paper is the external, thesis-structured statement of a doctrine developed and recorded inside a single production system; the origin system's internal research record grades each of its own claims against fetched literature and carries the evidence ledger this paper compresses. Origin-system claims herein are experience reports from one deployment, not measured results; literature claims carry their own citations. The author welcomes the only experiment that matters: an independent adoption.