The Loop, Verified

Internal study — evidence that Elis AI saves, contests, and selects decision chains.

Generated: 2026-08-08 02:36 UTC

The Loop, Verified: Evidence That Elis AI Saves, Contests, and Selects Decision Chains

Status: Internal draft — listed in the whitepapers index, which is restricted to sales reps and platform admins. Not a public document.
Version: 0.2-draft
Study dates: 2026-08-05 → 2026-08-06
Evidence policy (owner directive): every finding below rests exclusively on runs executed 2026-08-05 → 2026-08-06. No earlier telemetry is used anywhere in this paper.

---

1. Executive summary

Three claims, in decreasing order of evidential strength:

1. **The self-learning loop is operational end-to-end, verified live.** A user's 👎 on a real answer mints a rewritten decision chain as a traffic-gated canary; the canary serves deterministically alongside the incumbent; every run's grade is attributed to the exact chain version that served it; samples accumulate; and the promotion gate adjudicates. Every one of those transitions was observed on live runs and is reproducible from logged state.
2. **The selection gate works in both directions.** In the first two adjudicated contests in the system's history, the gate **promoted** a feedback-born chain that overtook its incumbent (0.7678 vs 0.7312 over 20 attributed samples) and **refused to promote** one that stayed worse (0.8838 vs 0.9252 at 10 samples). Promote-when-better and hold-when-worse, both demonstrated.
3. **Chain reuse is measurably cheaper.** On the one campaign with a clean first-ask baseline, repeat runs of the same question cost **15.4% fewer generation tokens** than the first ask (6,045 → 5,111 avg, n=29), consistent with the built-chain / cached-context hypothesis.

What this paper does **not** claim: a statistically powered accuracy improvement from learning (n=2 contests, one in each direction, self-graded, in a development environment), or any result on real tenant workload. §5 names the registered experiment that will settle the former and the pilot prerequisite for the latter.

---

2. Method

Protocol (per campaign): one fresh prompt, never seen by the system. Phase A: baseline runs mint the agent, chain, and first-ask token cost. Phase B: a 👎 with facets and a written note is submitted through the production feedback service, minting a canary rewrite. Phase C: the prompt is repeated in chunks of 5 (fresh conversation each run); the deterministic traffic gate splits serving between canary and incumbent; grades propagate to per-version EMAs; the promotion gate adjudicates at the sample threshold.

Controls and hygiene: chunked execution with a hard 90% host-memory ceiling; all state streamed to log files; exactly-once grade propagation via a cursor; mint throttle preventing version churn during contests. Test knobs, disclosed: canary traffic fraction 0.5 (prod default 0.10) and promotion threshold 10–20 samples, both set to reach verdicts in bounded run counts; neither changes gate *logic*.

Grader: the platform's own model-based grader. This measures the loop's ability to improve *its own quality signal* — the signal selection actually uses — not human-judged quality.

3. Results

3.1 The loop, segment by segment (all observed live)

- 👎 → canary mint through the production path: `user_feedback_rewrite` versions created at 0.5 traffic fraction in both completed campaigns.
- Deterministic serving: canary vs incumbent split reproduces from `(run_id, template_id)` hashing; serving and attribution agree independently.
- Attribution: 100% of orchestrator runs carry the exact chain version that served them (13/13, then sustained across ~60 campaign runs).
- Accumulation: per-version EMAs track grades in hand-checkable steps (e.g. 0.5 → 0.574 → 0.629 → 0.680 → 0.708 → 0.731 across five 0.8-grade runs).
- Self-repair: the cohesion sweep in quarantine mode archived 8 incoherent catch-all agents; a subsequent prompt in the archived agent's territory was re-served by a freshly minted specialist with no user-visible failure (gate-b verification). The evolution arm also minted an unprompted template upgrade from a high-quality run.

3.2 The contests

Campaign	Incumbent	Feedback canary	Samples	Verdict
Bakery (arithmetic reasoning)	0.7312	**0.7678**	20	**PROMOTED** — canary→active
Trains (multi-step word problem)	**0.9252**	0.8838	10	**HELD** — promotion refused; inside demote margin

The bakery trajectory is the learning story in miniature: the canary *trailed* at 5 samples (0.6881), led at 10 (0.7475), and won at 20 (0.7678) — evidence accumulated, the estimate converged, and the gate acted only at threshold. The trains hold is equally load-bearing: a plausible rewrite that graded worse was not promoted. A gate that only ever promotes proves nothing.

3.3 Token efficiency (first ask vs repeat)

Every dispatch is stamped with its run identity, so cost joins to the exact run that incurred it. On the trains campaign — the one with a clean first-ask baseline:

Phase	Runs	Avg generation tokens
First ask (runs 1–2)	2	6,045
Repeats (runs 3+)	29	5,111

−15.4% on reuse in tokens. Priced per dispatch at the serving model's catalog rate — tokens are not interchangeable across models, so each dispatch is costed at the model that actually ran it — the saving is larger in dollars:

Phase	Runs	Avg cost/run	Delta
First ask	2	$0.01084	—
Repeats	29	$0.00844	**−22.1%**

Dollars fall faster than tokens because reuse also shifts the *mix*: repeat runs lean harder on cheaper output-light dispatches and the small-tier model (§3.4). Repeat cost was also stable (bakery repeats: ~3.9k tokens ± small across chunks). Embedding-kind dispatches are reported separately by construction; indexing cost is never blended into inference cost. Catalog list rates used throughout (per 1M tokens in/out: haiku $1/$4, gpt-4.1 $1/$4,

o3-class $2/$8) — the same source the platform's savings feature prices from, never live billing.

3.4 Right-sized model selection (the binder's contribution)

Chain reuse (§3.3) is one savings mechanism; the runtime binder choosing the smallest adequate model is a second, and the campaign corpus measures it directly. Across all campaign runs (generation dispatches, run-joined):

Model	Calls	Share	Est. cost (catalog list)
claude-haiku-4-5 (small tier)	245	**96.1%**	$0.470
gpt-4o / gpt-4 / gpt-4o-search / gpt-3.5	10	3.9%	$0.012

96% of calls landed on a small-tier model while grader quality held at 0.73–0.93 — the definition of right-sizing: the binder did not buy frontier capacity these tasks didn't need.

Counterfactual: one frontier model doing every task in the corpus (same token volumes — 269,683 in / 53,965 out — priced at each model's catalog list rate) versus what the binder actually chose:

Scenario	Total cost	vs actual
Binder's actual mix (96% small-tier)	**$0.482**	—
All tasks on gpt-4.1	$0.486	+0.8%
All tasks on gpt-5	$0.825	**+71%**
All tasks on o3	$0.971	**+102%**
All tasks on claude-opus-4-1	$1.311	**+172%**

Sending this workload to an Opus-class frontier model would have cost **2.7×** what the binder's mix cost, for tasks a small model already answered at 0.73–0.93 grader quality. Honest bounds: the counterfactual holds token counts fixed, uses list prices only, and these campaign prompts are exactly the kind a right-sizer *should* route small — the claim is "small where small suffices, without quality loss," not a universal 63% saving. (gpt-4.1's list price sits at the small tier, which is why that comparison is flat — the saving comes from avoiding the models priced 2–4× higher.) A workload-weighted version over real tenant traffic belongs to the §5 milestones.

3.5 A finding the protocol produced for free

The third campaign (a code-generation prompt) minted its canary and then accumulated **zero** samples: code-shaped prompts route down an execution path not yet wired to the execution-record writer. The campaign protocol surfaces path-coverage gaps by construction — a run that produces no record is a run the loop cannot learn from, and the protocol makes that visible in one chunk.

---

4. What may be claimed, and what may not

Claimable now (each backed by logged, reproducible post-fix runs): the loop runs end-to-end; feedback causes chain rewrites; rewritten chains are contested under traffic gating; the gate promotes better chains and refuses worse ones; chain reuse reduced generation tokens 15.4% in the measured campaign; the matching layer self-repairs via cohesion quarantine without stranding demand.

Not claimable: that answers are *N% more accurate* because of learning (two contests, one per direction, is a demonstration, not a distribution); anything about real tenant outcomes (all evidence is first-party, dev environment, synthetic prompts); durability of the token saving beyond the measured campaign; human-judged quality (grader is model-based).

---

5. Registered next milestones

1. **Three-arm controlled experiment** (design pre-registered in the repo: control-repeat, treatment-rewrite, sham-mutation arms; paired within-question; blind grading; regression-to-mean floor reported). Its Phase 0 blocker — attribution — is now cleared. This is what converts §3.2 from demonstration to measurement.
2. **Wire the remaining early-exit paths** (expert, task-orchestrated, and the code path from §3.5) so every run enters the loop; then re-run the paused campaign.
3. **Real tenant corpus** — ≥2 external organizations, ≥1,000 org-scoped runs, ≥8 weeks, stable agent set. The pilot ask in the deck is the vehicle. No tenant-level claim before this exists.

---

6. Reproducibility

Raw chronological logs: `.claude/logs/learning-evidence/` — `promotion-campaign.log` (bakery contest, chunk by chunk, plus combined summaries), `campaign-trains.log`, `campaign-email.log` (path-gap finding),

plus the numbered query outputs from the 0.1 study retained for the §2 defect record. Deploy-facing summary: `.claude/docs/dev-17-deploy-notes.md`. Defect catalogue with fixes and verification: `.claude/docs/template-attribution-bug.md`. Experiment design: `.claude/docs/self-learning-proof-experiment.md`.

All measurements ran read-write against the development stack only; production was touched read-only and contributes nothing to the post-fix evidence base.
