VERILOOP CODER-E1: EVIDENCE-GOVERNED METHOD INHERITANCE FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-modifying coding agents can retain locally successful changes without establishing that those changes improve later error correction. We present VeriLoop, an evidence-governed framework that separates within-task verification from cross-task method inheritance. Its six operations organize evidence, falsification, inquiry, revision, verification, and verified internalization. A dual-gate protocol first validates a correction within its declared scope, then conditions inheritance on subsequent independent evidence, while retaining provenance and revocation paths. We instantiate the framework as VeriLoop Coder-E1 using a frozen Qwen3.6-27B backbone, four detachable Surface Host adapters, and an external Self-Harness. VeriLoop achieves 85.20% on SWE-bench Verified, 62.38% on SWE-bench Pro, 76.40% on Terminal-Bench 2.0, and 33.63% on DeepSWE. Three classification adapters improve on held-out capability endpoints; a fourth provides uncertainty estimates. Their LoRA matrices contain 3,481,640 parameters, or 0.012895% of the backbone. Mechanism audits cover 115 episodes and document selective admission and intact control boundaries. These findings support a practical, reversible method-inheritance architecture, but do not establish that inheritance causally improves downstream task success.

1 INTRODUCTION

Software-engineering agents combine language-model reasoning with repository navigation, tool execution, testing, and repair. Their performance therefore depends on the execution harness as well as the model (Jimenez et al., 2024; Yang et al., 2024). Recent work extends this harness from a fixed interface into a target of adaptation: experience can update strategy contexts, tool orchestration, and persistent rules (Zhang et al., 2026c; Lin et al., 2026; Zhang et al., 2026a). This creates a practical route to changing agent behavior without repeatedly updating a large backbone.

The central difficulty is deciding which changes deserve to persist. Success on the task that generated a revision can reflect an evaluator omission, task-specific overfitting, or random variation. If the same agent proposes a change, judges its success, and promotes it into future policy, a local error can become a persistent method. Self-correction limitations and evaluator preferences make this concern substantive (Huang et al., 2024; Panickssery et al., 2024; Gao et al., 2023). An auditable improvement process must separate candidate production from the authority to validate and inherit a candidate.

VeriLoop defines the recursive threshold in terms of future error correction: a validated revision must change how later independent tasks discover, judge, or repair errors, within a declared scope and with a path for revocation. This operational view draws on fallibilism and falsification (Peirce, 1877; Popper, 1962), but its contribution is a control protocol. Observations become admissible evidence only when their provenance and scope support a bounded state change. A verification gate authorizes local acceptance; a separate inheritance gate governs promotion into future method state.

Our contributions are threefold. First, we specify evidence admission and delayed method inheritance as distinct control decisions. Second, we implement them through a frozen backbone, detachable control adapters, and an external Self-Harness that records evidence and supports rollback. Third, we evaluate VeriLoop at the levels of system performance, capability endpoints, and mechanism integrity. These evidence levels are complementary but not interchangeable: an intact inheritance path is not by itself a causal demonstration of improved task success.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Feedback and adaptive harnesses. SWE-agent structures the agent–computer interface, while Reflexion stores verbal feedback across trials (Yang et al., 2024; Shinn et al., 2023). ACE adapts strategy context, AHE optimizes harness components, and Self-Harness proposes runtime changes from failed trajectories (Zhang et al., 2026c; Lin et al., 2026; Zhang et al., 2026a). VeriLoop focuses on the eligibility of a correction for future use: local verification, scoped inheritance, and revocation are represented separately.

Recursive improvement and evidence gates. The Darwin Gödel Machine and MetaSkill-Evolve explore changes to agent code or improvement skills (Zhang et al., 2026b; Wang et al., 2026). Proof-or-Stop binds lifecycle progression to execution evidence (Huang et al., 2026). VeriLoop connects such evidence constraints to method promotion rather than granting inheritance after a single passing result. Its confidence-bound formulation is related in motivation to high-confidence and safe policy improvement (Thomas et al., 2015; Laroche et al., 2019); it is an operational specification, not a new statistical guarantee.

3 VeriLoop: Evidence and Method Inheritance

3.1 State and evidence admission

At round t, the control state is $Z_t = (c_t, \sigma_t, H_t, \Omega_t, E_t, K_t, I, b_t)$. Here c_t is the claim, σ_t its scope, H_t competing hypotheses, Ω_t open evidence obligations, E_t admitted evidence, K_t the error-discovery, judgment, and repair mechanism, I preserved invariants, and b_t the remaining inquiry budget. The task anchor fixes the objective and acceptance boundary; revising it requires an explicit, versioned event.

An evidence obligation identifies an unknown whose possible outcomes can lead to different commitments. Let τ_o be the update associated with obligation o and π project the state onto effective commitments. An observation x is admitted as evidence when

$$\mathrm{Evidence}(x \mid o, Z_t) \iff \mathrm{Admissible}(x, o, \sigma_t) \wedge \pi(\tau_o(x; Z_t)) \neq \pi(Z_t). \tag{1}$$

Admissibility checks provenance, relevance, and scope. The update must be legal, nonempty, and bounded: it may change a hypothesis, action, inquiry direction, stopping condition, or method candidate. A retrieved passage or log that cannot affect such a decision remains context. Before execution, the harness records the claim, expected observation, potential outcomes, and cost; afterwards it binds actual receipts to the state transition.

3.2 Within-task correction and delayed inheritance

Figure 1 explains why the loop is a spiral. Its left-hand arrows allow failed verification to reopen falsification or inquiry, rather than forcing every revision forward. The outer dashed path indicates that an inherited method remains fallible in the next round. The right-hand gate distinguishes supported revisions from those requiring further tests. The lower-right inset contrasts a result reinforcing an unchanged method with a verified correction changing future practice. "Enter internalization" is a conceptual destination: in the dual-gate implementation, local verification produces only a provisional candidate, and later independent evaluation must establish inheritance eligibility. The spiral has six operations. *Evidence* establishes claims and boundaries; *Falsification* exposes observable failure conditions; *Inquiry* targets decisive unknowns; *Revision* changes the smallest sufficient failure boundary; *Verification* tests the correction and protected invariants; and *Verified Internalization* governs entry into future methods. These operations are not six mandatory model calls. Rejection, reopening, unresolved termination, and rollback are valid outcomes.

The verification gate G_V asks whether a revision r^* resolves the identified mismatch within its scope, survives relevant checks, and preserves invariants. Passing G_V can yield a local result or a provisional method candidate κ. It does not immediately modify the reusable method registry. The inheritance gate G_I asks whether the candidate changes later independent error correction with sufficient benefit and bounded regression risk.

$$\begin{aligned} G_I(\kappa) \iff\ & G_V(r^*) \wedge n \geq n_{\min} \wedge \neg\, \mathrm{OpEquivalent}(K^\kappa, K_t) \\ & \wedge\, \mathrm{LCB}_{1-\alpha}(\bar{d}_n) > \delta \wedge \mathrm{UCB}_{1-\alpha}(p_n) \leq \rho_{\max} \wedge \mathrm{Revisable}(\kappa). \end{aligned} \tag{2}$$

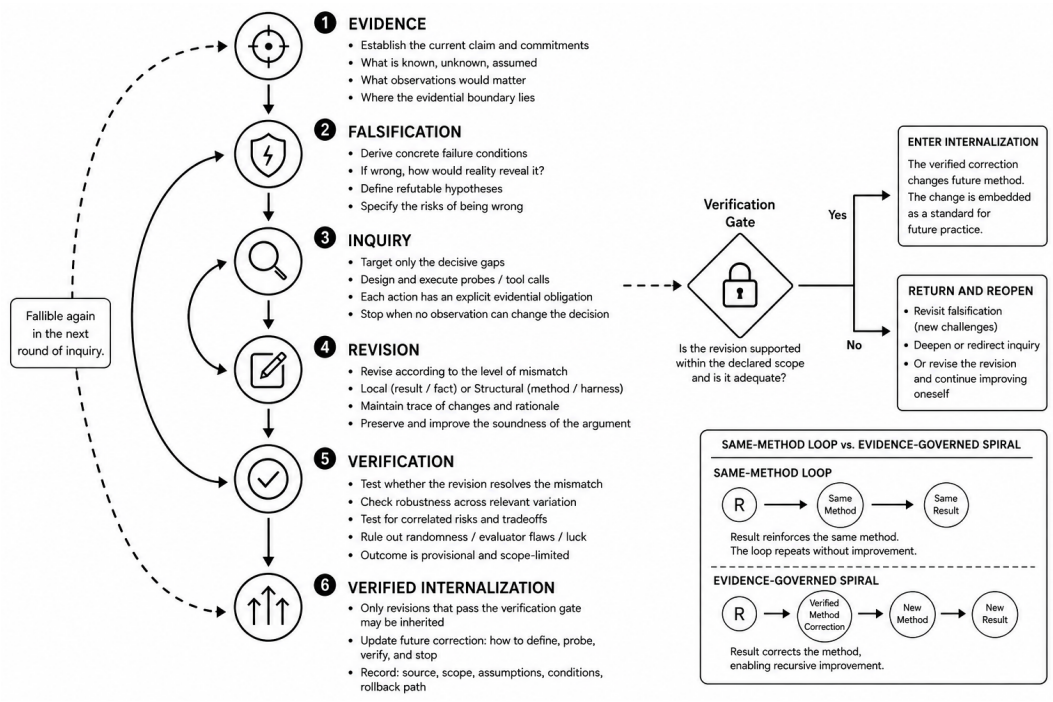


Figure 1: VeriLoop Spiral. Six operations connect a current claim to a revisable method. Return arrows reopen inquiry; the right inset contrasts repeating a method with correcting it. The depicted verification gate establishes local support; cross-task promotion additionally requires Equation 2.

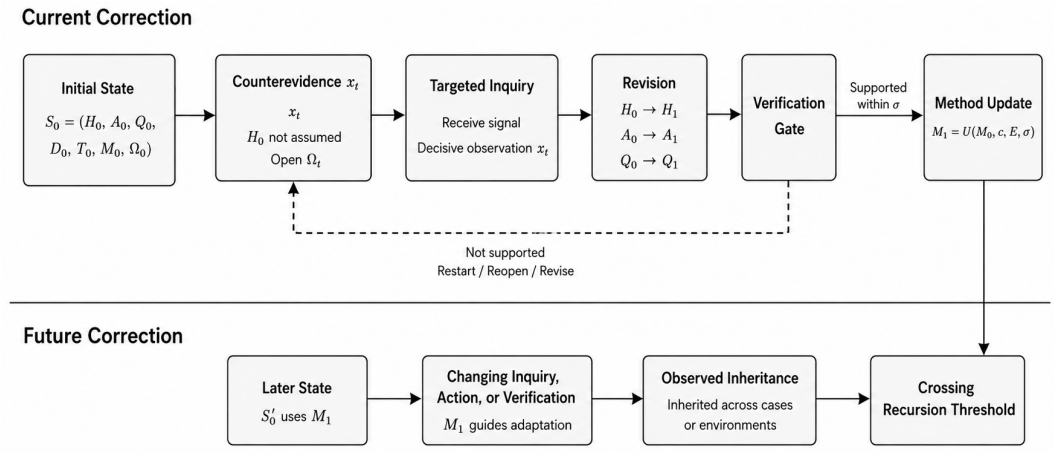


Figure 2: Current correction and later method effects. Local verification creates a scoped candidate; actual later effects must be evaluated before claiming recursive improvement. The diagram describes the control objective, not a measured gain curve.

Here $\bar{d}_n$ represents paired gain on later tasks, p_n the critical-regression rate, δ minimum gain, and $\rho_{\max}$ maximum permissible risk. Operational non-equivalence requires a change in discovery, judgment, or repair, not just rewritten text. Equation 2 specifies the conditions for method promotion.

A promoted candidate retains its trigger, refuted assumption, rule, supporting evidence, scope, failure signal, and rollback reference. New counterevidence can narrow scope, reduce priority, restore an earlier version, or reopen obligations. Thus inheritance remains conditional after promotion. Abstract rules may enter subsequent harness rules or PEFT data only under stable cross-task benefit and scope control; they are never written directly into the frozen backbone.

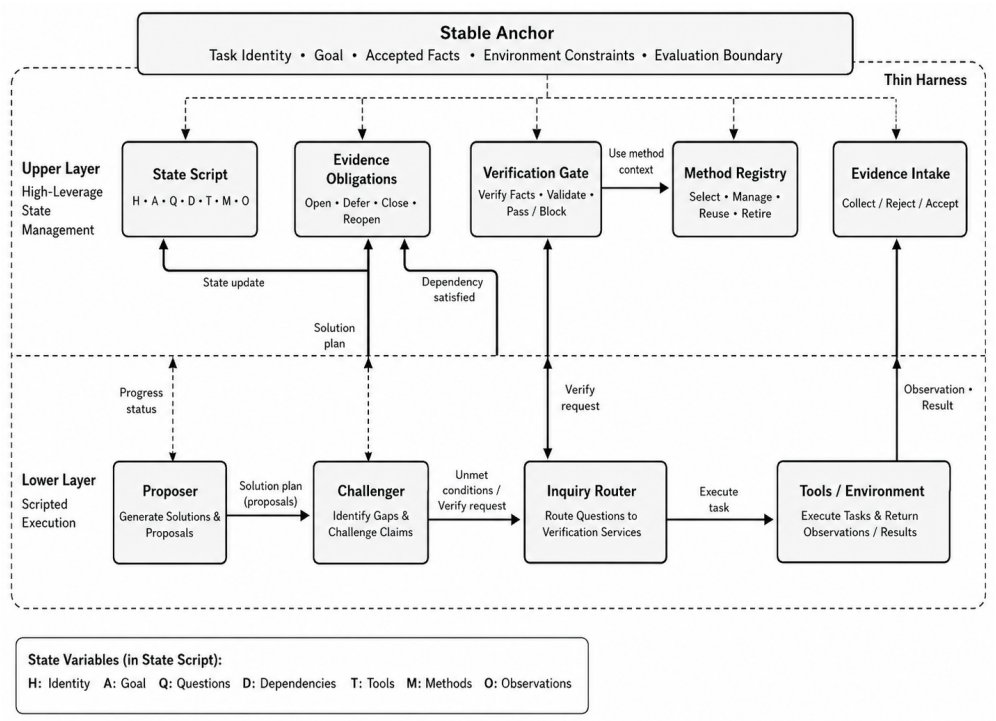


Figure 3: Self-Harness architecture. The stable anchor and upper control layer govern evidence, verification, and methods; the lower layer proposes, challenges, routes inquiries, and executes tools.

3.3 Frozen backbone and detachable control surfaces

VeriLoop Coder-E1 uses Qwen3.6-27B as a frozen code-reasoning backbone (Qwen Team, 2026). Four Surface Host adapters specialize tool specification, evidence binding, rollback, and uncertainty. ToolSpec supplies tool and parameter constraints; Evidence Binding links claims to sources and execution outcomes; Rollback predicts recovery needs and revision boundaries; Uncertainty estimates answer, evidence, execution, specification, and risk uncertainty.

The host reads the final valid-token representation from the last backbone layer, applies layer normalization, and maps it into capability-specific signals. Low-rank updates act on external `surface_host.*` mappings, not backbone attention or MLP projections. Signals are consumed by the harness rather than written back into logits or hidden states. Each surface can be loaded, replaced, or reverted independently. The reported 0.012895% parameter fraction counts only LoRA matrices, excluding task-specific heads and other trainable parameters.

3.4 Self-Harness execution

The model proposes hypotheses, plans, and code; the control plane maintains task contracts, evidence obligations, verification, and method promotion (Figure 3). Contracts encode permitted edits, interfaces, required and prohibited behavior, budgets, and invariants. Immutable stage artifacts retain parent references and content digests. Earlier rounds enter through structured references, limiting the ability of historical narration to overwrite the current contract.

Execution alternates implementation, machine verification, failure attribution, and targeted repair. Only actual tests, logs, exit codes, and artifact digests can constitute passing evidence. Failed checks become structured violations; the next repair receives the original contract, previous candidate, open violations, and protected invariants. Scores rank only candidates that already satisfy acceptance gates. Exhausted budgets produce an explicit safe stop and preserve recovery artifacts. Repository retrieval, external retrieval, and engineering-memory recall all share provenance, relevance, and scope checks.

3.5 ENGINEERING MEMORY AND OBSERVED INHERITANCE

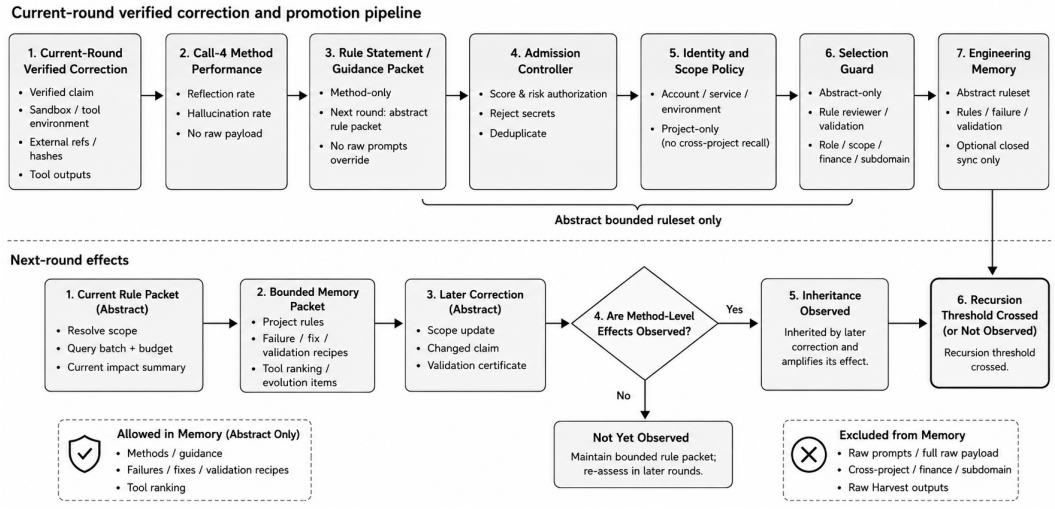


Figure 4: Rule admission and later effects. The upper pipeline abstracts, screens, and stores a verified correction under identity and scope constraints. The lower pipeline tests whether recalled rules change a later correction. Storage alone does not cross the recursive threshold.

Figure 4 expands the method-update path in Figure 2. In the upper pipeline, a verified correction is distilled by the method-performance stage into a method-only guidance packet and then screened by the admission controller. Deduplication and secret rejection are followed by identity, project-scope, and selection checks. Engineering memory accepts only bounded abstract rules, failure/repair recipes, and tool guidance, excluding raw prompts and full payloads.

In the lower row, a current rule packet and bounded recalled memory inform a later correction. The decision diamond asks whether a method-level effect is actually observed, such as a scope update, changed claim, or different verification behavior. The "No" branch retains a provisional packet for reassessment; the "Yes" branch records inheritance in later correction. The vertical connection from memory to the final status must therefore be read together with this observation test, not as automatic promotion upon storage. This is cross-task reuse within the approved scope: the depicted project-only policy does not establish unrestricted cross-project transfer.

4 EXPERIMENTAL SCOPE AND SETUP

The evaluation has three levels. System results cover four software-engineering benchmarks. Capability evaluation uses three before/after classification comparisons and one held-out uncertainty regression analysis, each with $n = 80$. Mechanism evaluation audits admission and control boundaries over 115 episodes, including 22 admitted episodes. These are different analysis units and should not be pooled into a single sample size.

Training uses 240 examples per classification surface and 220 for Uncertainty. Data combine harness trajectories and contract-derived samples with ToolACE, APIGen-MT-5k, sanitized MBPP, and non-Verified SWE-bench slices. SWE-bench Verified is explicitly excluded from training, although this does not constitute a comprehensive contamination audit of all four benchmarks. Complete source counts are available only for ToolSpec; mixture proportions for the other surfaces were not recorded.

All four runs use a single NVIDIA RTX PRO 6000 sequentially, a frozen 4-bit NF4 backbone with double quantization, BF16 computation, and host-side LoRA with rank 8 and scale 16, following the quantized-adaptation rationale of QLoRA (Dettmers et al., 2023). Appendix B gives the training configuration. Benchmark evaluations do not uniformly record task counts, per-task budgets, repeated-run seeds, complete evaluator versions, or interval estimates. Aggregate scores are therefore treated as descriptive results, not matched-compute causal comparisons.

5 RESULTS

5.1 END-TO-END SYSTEM PERFORMANCE

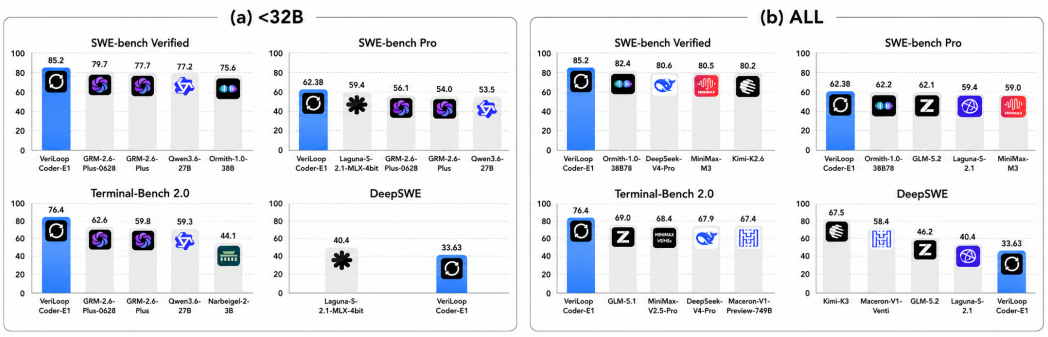


Figure 5: Public benchmark comparison as of July 27, 2026. Panel (a) is size-filtered and panel (b) is unrestricted by model scale. Each subpanel corresponds to one benchmark; blue bars denote VeriLoop Coder-E1.

Figure 5 provides two views of the same system: a size-filtered comparison on the left and an unrestricted comparison on the right. Within the displayed comparator sets, VeriLoop's blue bar is highest on SWE-bench Verified, SWE-bench Pro, and Terminal-Bench 2.0. DeepSWE shows the opposite boundary: 33.63 is below the displayed comparators, including the 40.4 score in the left panel. The figure therefore conveys both competitive results on three task families and a remaining challenge in the DeepSWE setting; it does not show uniform dominance.

Benchmark	Metric	Score (%)
SWE-bench Verified	Task resolution	85.20
SWE-bench Pro	Task resolution	62.38
Terminal-Bench 2.0	Pass rate	76.40
DeepSWE	Task resolution	33.63

Table 1: Complete-system results. Task distributions and protocols differ, so the table is not used for cross-benchmark ranking or causal component attribution.

5.2 CONTROL-CAPABILITY ENDPOINTS

Classification ($n = 80$ each)	Before	After	Δ	q_{BH}
ToolSpec	.1000	.4375	.3375	4.19×10^{-5}
Evidence Binding	.2750	.8250	.5500	3.35×10^{-7}
Rollback	.3500	.8625	.5125	3.35×10^{-7}
Uncertainty: held-out regression ($n = 80$), not a before/after comparison				
Answer RMSE= .147648; $R^2 = .2775$; RMSE skill= .1500				
$p = 1.30 \times 10^{-6}$; $q_{\mathrm{BH}} = 1.73 \times 10^{-6}$				

Table 2: Adapter endpoints and Holm-adjusted significance. Regression is presented separately from the before/after classification comparisons.

Evidence Binding and Rollback improve by 55.00 and 51.25 percentage points on their respective held-out classification endpoints. ToolSpec improves by 33.75 points but its final accuracy is only 43.75%, leaving substantial tool-selection error. Uncertainty supplies a five-dimensional continuous estimate; the answer dimension has $R^2 = 0.2775$. This is predictive evidence, not an estimated treatment effect of fine-tuning. Diagnostic metrics and raw p values appear in Appendix C.

The four surfaces contain 3,481,640 LoRA-matrix parameters and occupy 13.059 MiB in total. This demonstrates a small adaptation footprint for the measured control capabilities. It does not measure total training cost, all trainable parameters, or the computational overhead of the harness.

5.3 MECHANISM INTEGRITY

ID	Endpoint	Observed	p_{Holm}
M2	Admission–package coupling	$\phi = 1.0000$	4.46×10^{-15}
M3	Recursive-package closure	22/22	.01476
M4	Non-degenerate admission	22/115	2.30×10^{-7}
M5	Evolution-guard integrity	115/115	.00823

Table 3: Mechanism audit. M3 uses admitted episodes; M2/M4/M5 use episode-level records.

The admission rate is 19.13%, so the observed gate neither accepts every candidate nor rejects all candidates. All 22 admitted episodes produce a recursive package, and no evolution-guard violation is observed in 115 audited episodes. All four endpoints are statistically significant after Holm correction. These results concern admission, package transfer, and boundary compliance; they do not quantify a rule's performance benefit on an independent task. No observed violation is not equivalent to zero deployment risk.

6 DISCUSSION AND LIMITATIONS

The evidence supports three bounded conclusions: narrow control capabilities can be adapted with a small LoRA footprint; a runtime control plane can make method promotion auditable and reversible; and the complete configuration achieves the benchmark scores in Table 1. The missing connection is a controlled estimate of how much cross-task inheritance contributes to later task success. The current results therefore support the architecture and its measured components, not a general demonstration of recursive performance improvement.

Each capability endpoint has only 80 held-out examples. The mechanism audit is small, its episode dependence is not characterized, and the full multiple-testing family is not specified. Benchmark budgets and repeated-run variability are unrecorded. Missing training fields are treated as unknown rather than replaced with present-day defaults. Further, freezing weights does not ensure unchanged system behavior: adapters, retrieval, and memory can still create cross-task interference.

The framework is instantiated on one backbone. Some harness complexity may compensate for that backbone's planning and recovery limitations. Whether a stronger backbone permits a smaller control layer, and which constraints must remain external, are open empirical questions. Traceability and rollback constrain failures within implemented checks; they cannot guarantee that the checks cover all relevant errors.

7 CONCLUSION

VeriLoop separates local correction from eligibility for future method inheritance. A frozen backbone, detachable control surfaces, and an external Self-Harness implement evidence admission, scoped validation, and revocation. System, capability, and mechanism results support the feasibility of this design. Claims about causal downstream benefit remain contingent on direct cross-task comparison.

AI USE STATEMENT

In preparing this manuscript, the authors used generative AI tools, including OpenAI Codex, to assist with manuscript organization and drafting, Chinese–English translation, language editing, and the formatting of LaTeX, tables, and references. Generative AI was not used to generate datasets, implement the reported system, execute the reported experiments, or produce the reported measurements. The authors manually reviewed the AI-assisted text, equations, citations, figures, tables, and numerical values against the research records. The authors take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Sections 3–5 describe the control protocol, model and harness architecture, evaluation setting, and measured endpoints. Appendices A–D provide the control algorithm, training and supervision configuration, parameter accounting, statistical summaries, and operational rules. All aggregate values presented in the paper are retained with their stated analysis units and limitations. The current anonymous submission does not include the executable system implementation, raw trajectories, or per-example evaluation outputs; these omissions limit independent reproduction of the aggregate results.

REFERENCES

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023.

Jek Huang, Jeffery Hsia, Jiayi Sun, Freddie Shi, Wei Huang, and Ian H. White. Proof-or-stop: Don't trust the agent, trust the evidence—loop engineering for verifiable evidence-gated lifecycle control. *arXiv preprint arXiv:2607.14890*, 2026. doi: 10.48550/arXiv.2607.14890.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, and Ofir Press. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661, 2019.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026. doi: 10.48550/arXiv.2604.25850.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 68772–68802, 2024. doi: 10.52202/079017-2197.

Charles S. Peirce. The fixation of belief. *Popular Science Monthly*, 12:1–15, 1877.

Karl R. Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Basic Books, New York, 1962.

Qwen Team. Qwen3.6-27B. https://huggingface.co/Qwen/Qwen3.6-27B, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2380–2388, 2015.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. MetaSkill-Evolve: Recursive self-improvement of LLM agents via two-timescale meta-skill evolution. *arXiv preprint arXiv:2607.05297*, 2026. doi: 10.48550/arXiv.2607.05297.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a. doi: 10.48550/arXiv.2606.09498.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=pUpzQZTvGY.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=eC4ygDs02R.

A Control Specification and Interpretation

Figure 1 in the main text presents the six-stage control diagram. The evidence condition is a normative requirement on state transitions: provenance alone is insufficient when an observation has no decision consequence. Likewise, a changed action is insufficient when the supporting observation is inadmissible. The two checks jointly prevent arbitrary context accumulation from being counted as verified progress. An obligation $o = (u, \chi, Y, \tau_o)$ associates an unknown u, an inquiry specification χ, possible observations Y, and a state-update map τ_o. It is decisive if $|\{\pi_o(y; Z) : y \in Y\}| > 1$. If no falsifier exists, the protocol returns an untestable state. If no decisive observation is available, it returns an unresolved state. Neither outcome grants inheritance.

Revision levels are partially ordered by scope of impact. The protocol selects a minimal sufficient level that preserves task invariants and minimizes unsupported impact within that level. This objective differs from textual edit distance: a short patch can still alter behavior without evidence. The impact function is treated as a design criterion rather than a measured optimizer; its numerical implementation is outside the current evaluation.

Algorithm 1 Compact dual-gate control specification

Require: Claim c, scope σ, method K, invariants I, budget B, later tasks T^+
$Z \leftarrow \mathrm{Bound}(c, \sigma, K, I, B)$; $\kappa \leftarrow \varnothing$
while $\mathrm{RemainingBudget}(Z) > 0$ **do**
    Derive falsifiers; return Untestable if none exist
    Build decisive obligations; return Unresolved if none are executable
    $o \leftarrow \mathrm{Select}(\Omega)$; $x \leftarrow \mathrm{Observe}(o)$
    if $\neg\, \mathrm{Evidence}(x \mid o, Z)$ **then**
        continue
    end if
    $Z \leftarrow \tau_o(x; Z)$
    If obligations close without refutation, return Supported-Within-Scope
    If refuted, revise at a minimal sufficient boundary and verify
    On failed verification reopen; on result-only repair return the local result
    On verified method revision form provisional κ and leave the loop
end while
if $\kappa = \varnothing$ **then**
    return Unresolved
end if
for $q \in T^+$ **do**
    Evaluate paired (K, K^κ) and update gain/regression bounds
    Reject on hard counterevidence; promote only if $G_I(\kappa)$ passes
end for
If promotion is unsupported, retain Provisional status

The inheritance gate uses the risk constraint in Equation 2: the $(1 - \alpha)$ upper confidence bound on the critical-regression rate must not exceed $\rho_{\max}$. This notation is used consistently throughout the paper.

B Training and Adapter Records

Surface	Train	Eval	Supervision
ToolSpec	240	80	Tool choice, schemas, constraints, receipt requirements
Evidence Binding	240	80	Adjudication, source sufficiency, follow-up action
Rollback	240	80	Need, cause, revision mode, state protection
Uncertainty	220	80	Answer, evidence, execution, specification, risk

Table 4: Capability data and supervision.

ToolSpec's 320 examples comprise 202 harness trajectories, 34 boundary-repair samples, 53 ToolACE examples, 19 APIGen-MT-5k examples, and 12 sanitized MBPP examples. Evidence Binding uses non-Verified SWE-bench slices and sanitized MBPP as external supplements; Rollback uses non-Verified SWE-bench slices and MBPP failure-repair samples; Uncertainty uses MBPP and tool-ambiguity examples. Optimization used AdamW, cosine scheduling, warmup ratio 0.08, weight

Surface	Epochs	Steps	Best	Length	Learning rate
ToolSpec	2	60	2	1024	1.5×10^{-4}
Evidence Binding	2	60	2	1024	8×10^{-5}
Rollback	3	90	3	Unknown	8×10^{-5}
Uncertainty	3	84	2	1024	6×10^{-5}

Table 5: Training records. "Best" denotes best epoch; "Length" is maximum sequence length.

decay 0.01, and gradient clipping 1.0. FP16 was disabled. Counts, epochs, update steps, and best epochs are recorded in the training manifests.

For the final valid token t^*, the side signals are $h = H_L[t^*]$ and $z_k = g_k(\mathrm{LN}(h); \phi_k)$. Host mappings use $W'_k = W_k + (\alpha/r)B_kA_k$, with $r = 8$ and $\alpha = 16$. Neither the mappings nor their outputs are merged into backbone projections. Parameter counts are:

Surface	Pairs	LoRA params	Share	MiB
ToolSpec	11	901,120	.003337%	3.441
Evidence Binding	13	1,064,960	.003944%	4.066
Rollback	17	1,392,640	.005158%	5.317
Uncertainty	2	122,920	.000455%	.235
Total	43	3,481,640	.012895%	13.059

Table 6: LoRA-matrix parameters; other trainable host/head parameters are excluded.

C Statistical Summaries and Their Limits

Surface	Macro-F1	MCC	Raw p
ToolSpec	.4373	.3775	4.19×10^{-5}
Evidence Binding	.7845	.7614	1.65×10^{-7}
Rollback	.7530	.6890	1.67×10^{-7}
Uncertainty	N/A	N/A	1.30×10^{-6}

Table 7: Adapter diagnostics. Macro-F1 and MCC do not apply to continuous uncertainty regression.

For Uncertainty, mean MAE is 0.139382, weighted MAE 0.140354, mean RMSE 0.178999, and weighted RMSE 0.180485. Medium- and strict-tolerance accuracy are 0.7575 and 0.4600. Answer RMSE is 0.147648.

D Execution, Search, and Engineering Memory

The task anchor is $B = (I, O, F, C, V)$: task identity, objective, accepted facts, environmental constraints, and evaluation boundary. Contract compilation additionally records artifact type, target files, public interfaces, allowed edits, required behavior, verification depth, budgets, and permissions. Each stage receives an immutable artifact with stage, round, parent, digest, and SHA-256 references. State contains hypotheses, actions, inquiry direction, reasoning depth, stopping conditions, methods, and open evidence obligations.

ID	Threshold / reference	Raw p	Margin
M2	$\phi \geq .50$	8.93×10^{-16}	.5000
M3	Rate$>$ .80	.00738	.2000
M4	$[.05, .95]$	5.76×10^{-8}	.1413
M5	Rate$>$ .95	.00274	.0500

Table 8: Mechanism thresholds and raw significance. M2 also has MCC=1.0000.

An evidence transaction first matches an observation to an obligation. Unmatched observations stay context. Inadmissible observations are rejected and audited. An admissible observation produces a bounded candidate update; if the transition violates constraints, it is rolled back and recorded. Only a valid transition is committed as evidence. The system retains normalized receipts and references rather than unbounded raw dialogue. Inquiry continues only when an executable observation can change the decision and its expected value exceeds cost and risk.

Repository search resolves rules, entry points, callers, interfaces, tests, and build procedures according to current hypotheses. Web search is triggered by missing local evidence or time-sensitive dependencies and projects retrieved material into citable evidence atoms. Conflicting, stale, or unverifiable sources carry explicit risk flags. Memory retrieval removes identity, filenames, raw prompts, and answer-like content from previous tasks, retaining abstract engineering rules, failure patterns, verification recipes, and tool strategies.

Engineering memory stores provisional candidates after local verification. Promotion requires later changes in evidence obligations, edit location, tool choice, or verification behavior and renewed acceptance. Candidate provenance, scope, identity isolation, and contradiction checks govern storage and recall. Versioned revocation supports scope reduction, priority demotion, restoration, and reopening.

D.1 Fourteen operational disciplines

1. Prioritize the current request over historical memory and cached rules.
2. Escalate to research or repair workflows only with evidence or a contract gap.
3. Inspect entry points, callers, contracts, and failure signals before editing.
4. Produce the smallest complete, verifiable artifact satisfying the request.
5. Repair the violated invariant; use broader rewrites only when local repair is insufficient.
6. Judge task fulfillment beyond syntax and import checks.
7. Record failures, skipped steps, degradation, and unknowns explicitly.
8. Implement deterministic checks and parsing in typed code when feasible.
9. Treat `max_new_tokens` as a visible-output budget.
10. Give each tool call or recall triggering evidence and a downstream consumer.
11. Persist candidates, verification reports, selection reports, and progress.
12. Preserve interfaces, names, paths, repository style, and task invariants.
13. Deliver one selected artifact with the evidence package supporting it.
14. Load domain overlays only when required and keep them subordinate to the current task.