MONITORING IS NOT SELF-IMPROVEMENT: AN EVIDENCE LADDER FOR AGENT HARNESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent harnesses increasingly monitor execution traces, intervene during a run, and revise prompts, tools, or policies from experience. Evidence that any one of these mechanisms operates, however, does not establish that the agent has improved. We introduce an *evidence ladder* that separates three claims: measurement validity, runtime intervention efficacy, and persistent improvement on independent future tasks. Each level is expressed as an evidence contract with a distinct estimand, required observations, and invalid shortcuts. We instantiate the framework in HARNESSEVOLVE through a task-paired study of a two-check monitoring package on all 89 Terminal-Bench 2.0 tasks, comprising 267 matched task–condition evaluations per arm. The monitored harness obtains 60/267 verified successes versus 61/267 for the static harness (difference -0.4 percentage points; 95% CI $[-6.0, +5.2]$), while the four-rule operational endpoint also remains near zero (29.5% versus 29.8%). Applying the ladder shows why these outcome estimates cannot be replaced by mechanism telemetry: the checks have unequal exposure, the dominant operational event is outside their target set, and deterministic trace rules remain proxies whose semantic interpretation is limited by human annotation. Exploratory update studies separately verify proposal, gating, and deployment without treating execution of that loop as evidence of future-task improvement. The result is an auditable protocol for matching self-improvement claims to the experiments needed to support them.

1 INTRODUCTION

Monitoring is not self-improvement. An agent may expose a recognizable trace pattern without that pattern measuring a meaningful failure; a runtime check may change one episode without improving its outcome; and a system may deploy an edit without improving future tasks. Conflating these events turns evidence that a mechanism runs into a substantially stronger efficacy claim. The problem is especially acute for the *agent harness*—the prompts, tools, memory, and orchestration surrounding a base model—because harnesses are easy to modify and increasingly described as learning from their own experience.

We propose an *evidence ladder* for evaluating such claims and encode each level as an evidence contract specifying its estimand, required evidence, and invalid shortcuts. The first level, *measurement validity*, asks whether an operational signal measures the intended construct. The second, *runtime intervention efficacy*, asks whether an intervention has an opportunity to act, is taken up by the agent, and improves a specified outcome under controlled comparison. The third, *persistent improvement*, asks whether an update derived from prior experience improves performance on independent future tasks. Each level has a distinct estimand and failure mode; satisfying an earlier level cannot substitute for evidence at a later one.

Existing systems search over harness code, learn from execution traces, or adapt agents online (Lee et al., 2026b; Zhang et al., 2026; Park et al., 2026; Karten et al., 2026). Yet evaluations can blur detector activation, within-run behavioral change, and cross-task improvement. We operationalize the ladder in HARNESSEVOLVE, a controlled evaluation scaffold that records operational trace events, applies specified runtime checks, and proposes regression-gated harness updates. The main experiment targets the runtime-efficacy tier through a two-check package—a repeated-command warning and a verification gate—while the update loop is assessed separately at the persistence tier.

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.

Across all 89 Terminal-Bench 2.0 tasks (Merrill et al., 2026) and three task-matched conditions, the monitored harness obtains 60/267 verified successes, compared with 61/267 for the static harness. The paired estimate is -0.4 percentage points (95% CI $[-6.0, +5.2]$); the interval crosses both zero and the -5 pp non-inferiority margin, and the operational endpoint shows a similarly near-zero difference. The evidence ladder decomposes this package-level estimate into measured intervention exposure, target alignment, and construct-validity evidence, without attributing the estimate to a single mediator. Separate exploratory studies verify end-to-end update execution, while improvement on independent future tasks remains the next evidence tier to establish.

This paper makes three contributions. First, we encode each level of the ladder in an evidence contract that assigns measurement validity, runtime efficacy, and persistence distinct estimands, falsifiable evidence requirements, and invalid shortcuts. Second, we conduct a task-paired controlled evaluation with explicit estimands, task-clustered uncertainty, and treatment-uptake analysis. Third, we connect construct validity, intervention exposure, target alignment, and independent-task transfer into a diagnostic protocol for identifying the evidentiary link that limits a self-improvement claim. Together, these contributions provide a reusable protocol for calibrating self-improvement claims to the evidence supplied by a particular experiment.

2 Related Work

Harness optimization and persistent updates. A growing literature treats the harness, rather than model weights alone, as an optimization target. Meta-Harness selects proposals using a labeled validation split (Lee et al., 2026b); Task-CoEvolve, HarnessLens, and HarnessBank develop lower-cost evaluation, verification, or search procedures for related loops (Miyai et al., 2026; Xu et al., 2026; Luo et al., 2026). Other systems derive updates from execution evidence. Self-Harness mines verifier-grounded failure signatures and regression-tests candidate edits (Zhang et al., 2026); AutoSaddler diagnoses traces and validates code-level patches (Park et al., 2026); and RHO and RHI refine prompt-level specifications from re-solved trajectories or recursive feedback (Pan et al., 2026; Lee et al., 2026a). Continual Harness studies online refinement in embodied domains (Karten et al., 2026), while EvoHarness-RL and SIA combine harness adaptation with learned policies or test-time model updates (Ning et al., 2026; Hebbar et al., 2026). These works establish that harnesses are an important optimization surface. They also motivate a sharper question: which part of a self-improvement claim is actually identified by a given evaluation?

Runtime monitoring, recovery, and process-level evaluation. Runtime systems diagnose failures and attempt repair within an ongoing interaction. AgentTether adds a dependency-aware diagnosis-and-repair layer (Zhao et al., 2026); Self-Harness, AutoSaddler, HarnessBank, and Continual Harness also connect execution signals to recovery or adaptation (Zhang et al., 2026; Park et al., 2026; Luo et al., 2026; Karten et al., 2026). Work on smaller-model harness adaptation further shows that orchestration can change the cost–quality tradeoff (Yang et al., 2026). Table 1 organizes the closest systems by the empirical links they report. The table is not a quality ranking: it distinguishes mechanism coverage from designs that can identify measurement validity, within-run efficacy, or transfer.

The comparison reveals a recurring asymmetry. Many systems implement an update or recovery path and evaluate whether it executes; fewer evaluations separate the validity of the triggering signal, the effect of the within-run action, and the transfer effect of a frozen update. Held-out validation can select a patch without establishing that the patch improves a future task distribution, while a repair rate on initially failed trajectories conditions on a different estimand from the intent-to-treat effect of deploying the repair layer. Our evidence ladder makes these distinctions explicit. The controlled study identifies the package-level runtime effect; construct annotation and uptake measurements diagnose earlier links; persistent-update studies address the next rung separately.

Failure taxonomies and operational measurements. Agent-failure studies provide semantic taxonomies for interpreting trajectories. MAST catalogs failures in multi-agent traces, including premature termination and incomplete verification (Cemri et al., 2025); GAE organizes failed sessions across agent architectures into a finer-grained error taxonomy (Bandel et al., 2026); and TRAIL localizes errors across reasoning, coordination, and system execution (Deshpande et al., 2025). Our four rules instead define reproducible trace events associated with working-set overload, command-

Table 1: Reported coverage in selected harness-optimization systems. ✓indicates explicit coverage, ·partial coverage, and × no coverage in the audited cited version. Columns record whether the work implements a measurement or diagnosis component (Meas.), a within-run action (Run), or a persistent harness update (Upd.); evaluates held-out or disjoint tasks (Ind.); reports resource matching or accounting (Res.); declares an analysis plan fixed before the relevant results (Pre.); and reports null, adverse, or regression-rejection outcomes (Null). Symbols describe reported coverage, not empirical validity. Coding scope and qualifications are given in Appendix G.

Work	Meas.	Run	Upd.	Ind.	Res.	Pre.	Null
Meta-Harness[a]	×	×	·	·	✓	×	·
Self-Harness[b]	✓	·	✓	·	×	×	·
AutoSaddler[c]	·	×	✓	✓	·	×	✓
Continual Harness[d]	·	✓	✓	×	·	×	✓
Task-CoEvolve[e]	×	×	✓	·	·	×	✓
HarnessLens[f]	·	×	✓	✓	·	×	✓
HarnessBank[g]	✓	✓	✓	✓	·	×	✓
EvoHarness-RL[h]	×	✓	✓	·	×	×	·
RHI[i]	×	×	·	×	×	×	✓
SIA[j]	·	×	·	·	×	×	×
AgentTether[k]	✓	✓	×	·	×	×	✓
HARNESSEVOLVE	✓†	✓	✓	·‡	·	·⋆	✓

Sources a–k correspond to the cited systems in table order. †Deterministic operational proxy with limited human construct validation. ‡Persistent-transfer evidence is exploratory. ⋆Endpoints and analysis plan were fixed internally before the relevant analyses, without external preregistration.

Table 2: Evidence contract for agent-harness improvement. Each level requires its own estimand; the shortcut in the fourth column is not valid evidence for that level.

Claim	Estimand	Required evidence	Invalid shortcut
Measurement	Proxy–construct agreement	Construct labels assigned blind to detector output under a stated sampling design	A rule fires, therefore the failure exists
Runtime efficacy	Controlled effect on a stated outcome	Opportunity, exposure, uptake, and a matched outcome comparison	Behavior changed, therefore performance improved
Persistence	Effect on independent future tasks	A frozen update and controlled transfer evaluation	An edit was accepted or deployed, therefore the system improved

surface churn, uncontrolled retries, and premature completion, with some rules partly depending on terminal outcomes. We report rule incidence separately from official verified success and use human annotation to delimit semantic interpretation. The resulting empirical claims are scoped to the specified detector and two-check package; semantic failure prevalence and monitoring efficacy in other settings require their own evidence.

3 METHOD

3.1 AN EVIDENCE CONTRACT FOR SELF-IMPROVEMENT CLAIMS

Let h denote an agent harness around a fixed base model, τ a task trajectory, and $Y(\tau) \in \{0, 1\}$ an externally verified task outcome. We separate three claims that are often grouped under "self-improvement." Table 2 states the corresponding evidence contract.

For operational mode m, a deterministic rule maps a trajectory to $d_m(\tau) \in \{0, 1\}$. For arm a with N_a classifiable trajectories,

$$I_m(a) = \frac{1}{N_a} \sum_{i=1}^{N_a} d_m(\tau_{ia}), \qquad I_{\text{macro}}(a) = \frac{1}{4} \sum_m I_m(a). \tag{1}$$

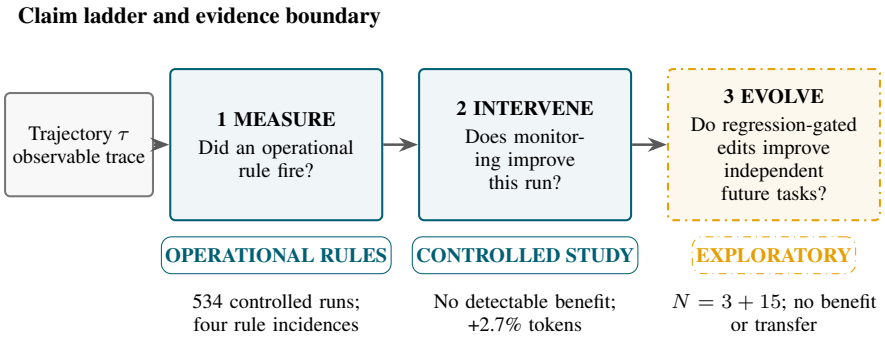

Figure 1: **The evidence ladder separates three empirical claims.** Operational signals require construct validation; runtime interventions require exposure, uptake, and controlled outcome effects; persistent improvement requires transfer to independent future tasks. The controlled study contains 534 runs forming 267 matched task–condition cells per arm.

Table 3: Observable intervention contract. Opportunity defines where a check can act; exposure records where it did act; uptake is an immediate post-trigger response; and efficacy is assessed by the matched external outcome, not by trigger or uptake alone.

Check	Opportunity and action	Uptake measurement	Targeted event
Repeat warning	A normalized command fails twice with a diagnostic; append a prompt to diagnose before repeating	Next command and any later retry event	Uncontrolled retry
Verification gate	A completion declaration occurs within the implemented 15-turn gate horizon; request an objective verification command before termination	Further command after the gate and terminal verifier outcome	Premature completion

These quantities are reproducible measurements of rule incidence; they become measurements of semantic failure only if construct validity is established. For the controlled experiment, the principal outcome estimand is $\Delta_{\text{succ}} = p_{\text{mon}} - p_{\text{static}}$. Because lower operational incidence is preferred, we report $\Delta_{\text{det}} = I_{\text{macro}}(\text{static}) - I_{\text{macro}}(\text{mon})$. Positive values therefore favor monitoring for the detector endpoint, whereas positive values of Δ_{succ} favor monitoring directly.

3.2 Controlled Instantiation: Two Runtime Checks

The controlled instantiation compares a static harness with the same harness plus two runtime checks. A *repeat warning* is appended when a normalized command fails twice with a diagnostic, asking the agent to diagnose rather than repeat the command. A *verification gate* is issued when the agent first declares completion within the implemented 15-turn gate horizon, requesting an objective verification command before the run can end. The run itself may continue for up to 25 turns. Both arms otherwise share the base model, system prompt, tools, verifier, and resource-control rules.

The runner records model replies, parsed commands, command outcomes, termination status, verifier reward, and resource use. We operationalize four trace events—working-set overload, command-surface churn, uncontrolled retry, and premature completion—as deterministic rules (Appendix B). The intervention target map is explicit: the repeat warning targets retry and the verification gate targets premature completion, while overload and command-surface churn remain outside the target set. Table 3 states the observable contract for each check. This mapping prevents a change in an untargeted component of a composite endpoint from being interpreted as evidence for or against the check's intended mechanism.

Table 4: Primary controlled outcomes under the shared resource-control rules. The success effect is monitored minus static; the incidence effect is static minus monitored, so positive values favor monitoring in both rows. Intervals use 89 task clusters.

Outcome	Static	Monitored	Estimated effect [95% CI]
Verified success ↑	61/267 (22.8%)	60/267 (22.5%)	-0.4 [$-6.0, +5.2$] pp
Operational incidence ↓	29.8%	29.5%	$+0.3$ [$-2.1, +2.8$] pp

For run i, let O_i denote an intervention opportunity, Z_i observed exposure, B_i the immediate post-trigger behavior, D_i the targeted operational event, and Y_i verified task success. These variables form an evidence chain rather than interchangeable outcomes. Opportunity determines the population on which a check could operate; exposure establishes treatment fidelity; uptake describes what follows a trigger; the target event assesses the intended operational pathway; and Y_i supplies the task-level criterion. The primary intent-to-treat comparison remains defined over all matched runs. Conditioning on $Z_i = 1$ is useful for diagnosis but is post-treatment and does not identify the causal effect of triggering.

3.3 Persistent Update as a Separate Claim

Given trajectories from completed tasks, an editor can propose a bounded harness change Δh. A fixed regression procedure accepts or rejects the candidate before it is used on subsequent tasks; the evaluation verifier, security boundary, and regression criteria are outside the update surface. This procedure establishes a bounded retention mechanism and isolates persistence as a separate estimand: an accepted, frozen update must outperform the untouched harness on independent future tasks under matched evaluation conditions. Our persistent-update studies provide exploratory instantiations of this sequence and are analyzed separately from the controlled runtime intervention.

4 Experiments

4.1 Experimental Setup

The controlled study uses all 89 Terminal-Bench 2.0 tasks and their official deterministic verifiers. Both arms use **glm-5.3-flash**, the same base harness and tools, and common resource rules: a 100K pre-call token threshold, 25 turns, 2K output tokens per turn, and 8K characters of observation context. Because the token threshold is checked before a model call, realized totals can exceed 100K. The monitored arm adds the two checks in Section 3.2. Archived records group runs into three nominal configuration batches labeled 0.2, 0.5, and 0.8. Request payloads are unavailable, so these labels define matched task–batch cells rather than verified decoding-temperature manipulations. There are 267 matched cells per arm and 534 controlled runs in total. Success is verifier reward 1; zero, missing, or invalid rewards count as failure.

We estimate paired arm differences and obtain 95% intervals by bootstrapping the 89 tasks as clusters, preserving the three conditions within each task. The planned criteria were a 20% relative reduction in four-rule macro incidence (5.96 percentage points) and verified-success non-inferiority within -5 percentage points. Detector definitions, deviations from the initial analysis plan, supplementary tests, and descriptive comparator arms are reported in Appendix A.

Results at a glance. Table 4 shows near-zero estimates for both endpoints. The success interval crosses both zero and the -5 pp non-inferiority margin, while the incidence interval excludes the planned 5.96 pp reduction for this configuration. The remaining analyses use the evidence ladder to organize four distinct evidence links: operational-measure validity, treatment exposure and uptake, controlled task outcomes, and transfer of persistent updates.

4.2 RQ1: Do the Operational Rules Measure the Intended Failures?

Applying the four rules to all controlled trajectories gives static/monitored incidence rates of 0.038/0.023 for overload, 0.483/0.517 for command-surface churn, 0.524/0.479 for retry, and

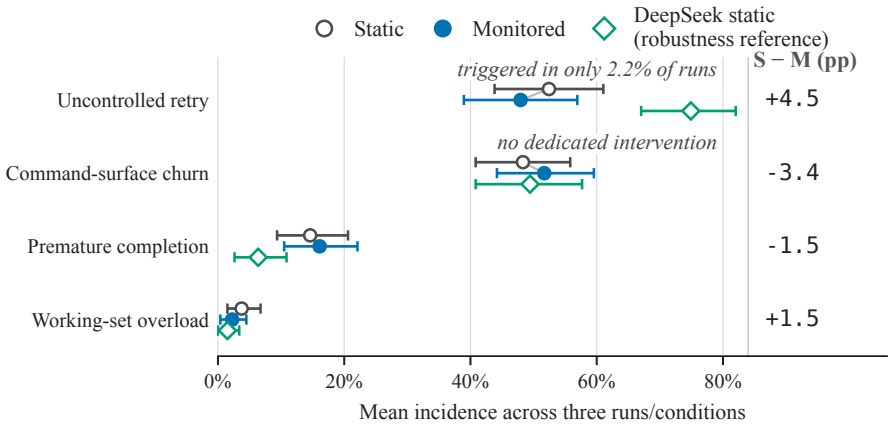

Figure 2: **Operational-flag composition differs between the static and monitored arms while the estimated macro-average incidence difference remains near zero.** Static/monitored comparisons are controlled; the DeepSeek arm is descriptive. The dominant command-surface-churn proxy lies outside the intervention target set. Per-mode values are descriptive decompositions.

Table 5: Observed exposure and post-trigger behavior in the monitored arm. Counts are diagnostic measurements, not causal effects.

Check	Exposed runs	Immediate response	Downstream observation
Repeat warning	6/267	1 different next command	0/6 later retry events
Verification gate	63/267	57/63 issued another command	30/63 failed; 27/63 retained flag

0.146/0.161 for premature completion. These are descriptive decompositions of the operational endpoint, not separate confirmatory outcomes.

A construct-validity study sampled with replacement 100 trajectory segments representing 85 unique task–arm–condition cells. Two internal annotators labeled each segment independently and a third adjudicated disagreements. Inverse-multiplicity weighting of repeated cells did not change the qualitative conclusion. Four rows recorded prior detector-label exposure for one annotator, so strict detector blindness was not established for those rows. Inter-annotator agreement reached $\kappa \leq 0.49$, and the adjudicated semantic-positive counts were too small for stable rule-level semantic calibration. Because the rules are deterministic, they provide reproducible operational measurements; the annotation results limit their semantic interpretation, leaving semantic-failure prevalence unestimated. Sampling, blinding, labels, and rule-level results appear in Appendix C.

4.3 RQ2: What Exposure and Post-Trigger Responses Are Observed?

The observed exposure–uptake funnel differs sharply across the two checks (Table 5). The repeat warning fired in 6/267 monitored runs. No further retry event was observed in those six runs, and one issued a different next command. The verification gate fired in 63/267 runs, while 30 additional completion declarations were not gated under the implemented eligibility rule. Following the trigger, 57/63 gated runs issued a further command. These observations show that the checks were executable and, especially for the gate, often elicited continued interaction. They do not by themselves establish outcome efficacy.

The exposure profile also clarifies endpoint alignment. Command-surface churn, the largest component of the four-rule composite, lies outside the two checks' target set; overload is likewise untargeted. Thus the macro endpoint tests a broad operational package effect, whereas the retry and

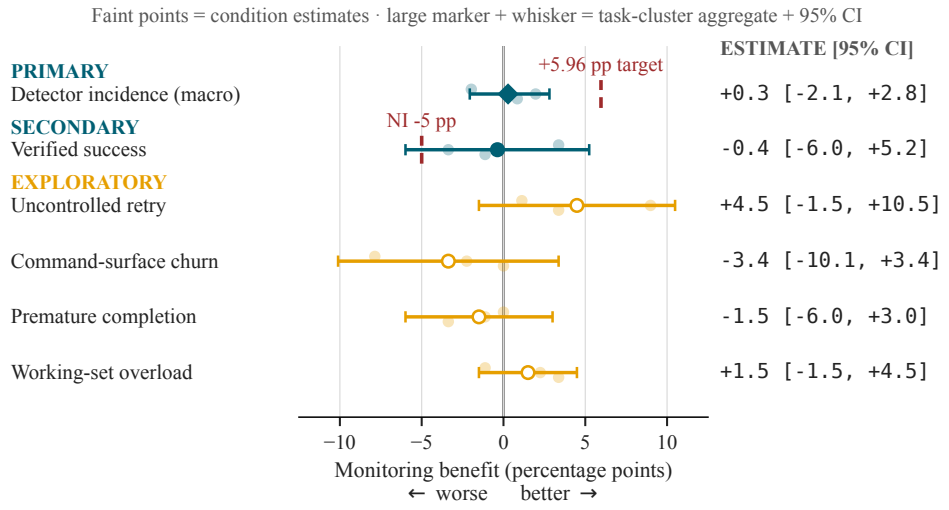

Figure 3: **Controlled effects of the two-check monitoring package.** Positive values favor monitoring (monitored minus static for success; static minus monitored for incidence). The success interval crosses zero and the -5 pp non-inferiority margin, while the incidence interval excludes the planned 5.96 pp reduction. Markers show paired estimates with 95% task-cluster bootstrap intervals over 89 tasks; per-mode rows are descriptive.

premature-completion components are the direct mechanism targets. The complete opportunity–trigger–uptake funnel, including late declarations, is reported in Appendix D.2.

Under the shared pre-call threshold, mean token consumption was 78.1K for the monitored arm and 76.1K for the static arm, a descriptive difference of 2.7%. Across both arms, 249/534 runs crossed 100K after their final admitted call; the maximum realized total was 117,827 tokens. This comparison was not a separately powered endpoint.

4.4 RQ3: Does Monitoring Improve Task Outcomes?

The static arm solves 61/267 cells and the monitored arm solves 60/267. Both solve 38 cells, static alone solves 23, monitored alone solves 22, and neither solves 184. The paired monitored-minus-static effect is -0.4 pp (95% CI $[-6.0, +5.2]$). The interval crosses zero and the -5 pp margin, leaving both a success gain and non-inferiority unestablished. Exact task-cluster sign-flip tests agree with the bootstrap analysis. Post-hoc mode-set, threshold, triggered-subset, and nominal-batch analyses preserve this package-level conclusion (Appendix E).

The operational endpoint similarly has a near-zero estimated effect: macro incidence is 29.8% for static and 29.5% for monitored, giving $\Delta_{\text{det}} = +0.3$ pp (95% CI $[-2.1, +2.8]$). The estimate corresponds to a 0.9% relative reduction, compared with the planned 20% target. Together with the success estimate, this bounds the package-level effect under the evaluated model, benchmark, and budget: the data exclude the planned incidence reduction and do not detect a verified-success gain.

4.5 RQ4: Do Harness Updates Persist and Transfer?

Across four exploratory studies, the update pathway completes proposal, regression testing, retention, and later-task evaluation. The corresponding comparisons yield 0/3 in the mechanism study, 2/15 versus 3/15 in the same-order study, 4/15 versus 4/15 in disjoint transfer, and 6/29 versus 7/29 for the Self-Harness reimplementation (Zhang et al., 2026). These results establish end-to-end mechanism execution while leaving persistent performance gains unestablished. Appendix F provides the full designs and outcomes.

Table 6: Minimum reporting checklist implied by the evidence contract. The rightmost column names the conclusion that the listed evidence does not, by itself, justify.

Claim	Minimum report	Invalid inference
Measurement validity	Rule definition, sampling frame, annotation procedure, agreement, and proxy–construct calibration	Reproducible firing implies semantic validity
Runtime efficacy	Eligibility, exposure, uptake, target alignment, matched external outcome, and uncertainty	A trigger or behavior change implies improvement
Persistent improvement	Frozen update, untouched control, independent future tasks, and matched evaluation conditions	Acceptance or deployment implies transfer
Resource fairness	Actual consumption and stopping behavior for every arm, not only nominal limits	A shared configuration implies equal realized cost

5 DISCUSSION

A package-level null can still resolve an important empirical question. The controlled comparison estimates the effect of deploying the evaluated two-check package, not the value of monitoring in the abstract. Under the specified model, harness, benchmark, and resource rules, the package does not produce a detected verified-success gain or the planned reduction in the broad operational endpoint. This result is informative because successful execution of a monitor, repair path, or update pipeline is otherwise easy to mistake for improvement. A matched comparison converts that implementation claim into a falsifiable outcome claim and supplies uncertainty wide enough to distinguish "no detected gain" from equivalence.

The evidence ladder separates outcome effects from evidence-chain diagnostics. The package estimate is accompanied by measurements of distinct links: deterministic operational rules whose semantic interpretation remains limited, observed intervention exposure, a dominant endpoint component outside the target set, and post-trigger behavior alongside verified success. These links answer different questions. Trigger counts establish that treatment was delivered; continued commands establish a form of uptake; target-event counts indicate whether the intended operational pattern remains observable; and the verifier measures whether the task was solved. None substitutes for another. The decomposition therefore localizes what a stronger follow-up design must change without assigning a retrospective single cause to the aggregate result.

Opportunity, exposure, uptake, and outcome form a complete reporting funnel. Runtime interventions are unusually vulnerable to dilution: a check may be available in every run but eligible in few, trigger in fewer, and alter behavior without changing the terminal outcome. Reporting only aggregate success hides this implementation structure, while reporting only triggered cases conditions on a post-treatment variable. The full funnel makes both errors visible. For causal mechanism attribution, a future study should randomize warning content, gate timing, or eligibility at the opportunity level; for deployment decisions, the intent-to-treat package effect remains the relevant quantity.

Reproducibility and construct validity are orthogonal. A deterministic rule can be reproduced exactly while remaining an operational proxy for the intended semantic phenomenon. Human disagreement can reveal an underspecified construct as well as detector noise. Operational endpoints should therefore be named by what they compute, calibrated against independently sampled labels when semantic interpretation matters, and kept distinct from task outcomes. Target alignment is equally important: a composite endpoint represents an intervention's mechanism target only to the extent that its components lie within the target set. Broad composites may still be useful as system-level summaries, but component-wise reporting is necessary for mechanism interpretation.

Table 6 turns the ladder into a reporting protocol. The rows are deliberately separable: a study can provide strong runtime evidence while leaving persistence open, or establish a reproducible operational measure without validating its semantic label. This modularity supports calibrated claims rather than a binary verdict on whether an agent "self-improves." It also makes negative results cu-

mulative: a null package effect accompanied by exposure and alignment measurements identifies a more precise redesign target than an outcome number alone.

Persistent improvement is a separate evidence level. Proposal, testing, acceptance, and deployment establish mechanism operation and characterize the retention gate. Persistent improvement is established by a frozen update, a matched untouched-harness comparison, and gains on independent future tasks under controlled resources. This distinction applies whether an update is produced by search, reflection, preference learning, or code editing: the generation mechanism changes, but the transfer estimand does not. In this study, the evidence contract bounds the runtime conclusion to the evaluated package and identifies independent future-task transfer as the next evidence level to establish. More generally, the same ladder strengthens positive results by making clear which empirical claim each experiment supports.

6 LIMITATIONS

Measurement validity. The four rules are operational trace-event proxies rather than validated semantic failure modes. The construct-validity study uses 100 annotation instances (85 unique task–arm–condition cells), two internal annotators, and a third internal adjudicator. Agreement is at most moderate and adjudicated positives are sparse, so recall and prevalence of the intended semantic constructs cannot be estimated reliably. The composite also mixes trace-derived rules with rules that use terminal outcomes. Its rates should be interpreted as incidence under the stated definitions, not as a general failure taxonomy.

Causal and statistical scope. The controlled comparison estimates the intent-to-treat effect of the complete two-check package. It does not separately identify effects of warning content, gate timing, added context, or post-trigger behavior, and triggered-subset comparisons are post-treatment descriptions rather than causal estimates. The three nominal conditions are archived batch labels rather than independent random seeds, and request-level decoding parameters are unavailable. Inference therefore clusters at the 89-task level; it neither attributes between-batch variation to temperature nor estimates seed-level variability. Detector definitions were fixed before mode-level analysis but after aggregate success was inspected and were not externally preregistered. Deviations and post-hoc analyses are disclosed in the appendix.

External validity and persistence. Both controlled arms use one base model, one harness, Terminal-Bench 2.0, and a 100K pre-call token threshold. Results may differ with other models, tools, task distributions, budgets, or interventions aligned to different events. No equal-budget human-tuning baseline is included. The persistent-update studies are small and exploratory; their non-improving comparisons neither establish benefit nor rule out benefit in a stronger design. Accordingly, our evidence supports a bounded conclusion about this package and a general evaluation protocol, not a universal limit on monitoring or agent self-improvement.

7 CONCLUSION

Claims of agent self-improvement should be decomposed into what is measured, what changes during the current run, and what improves on independent future tasks. We formalized this decomposition as an evidence contract and applied it to a task-paired evaluation of two runtime checks. In the controlled model–harness–benchmark configuration, monitoring yielded a near-zero verified-success estimate whose interval crossed both zero and the planned non-inferiority margin, together with a similarly near-zero operational endpoint. The evidence contract complements this package-level estimate with measurements of intervention exposure, target alignment, and construct validity. Exploratory update studies establish end-to-end edit proposal, gating, and deployment, with independent future-task efficacy as the next evidence tier to establish. Together, these findings motivate an auditable protocol for evaluating self-improvement claims: test measurement validity, establish intervention exposure and controlled outcome effects, and evaluate frozen updates on independent future tasks. The present study completes only the evidence links supported by its data and leaves the remaining links explicit. This separation makes self-improvement claims auditable by tying every conclusion to the evidence it requires.

Reproducibility Statement

The supplementary materials are an anonymous, self-contained package with the aggregate result tables and analysis code behind the reported numbers. The paired exact tests, the success-rate and forest figures, and the underlying summary statistics regenerate offline from the packaged tables, byte-for-byte; the trajectory-level analyses are provided as frozen aggregate records. Appendix H describes the corresponding reproducibility scope.

Operational detector classifications require the underlying command and event sequences. Raw trajectories are not distributed under the applicable benchmark, provider, and security constraints. The released materials therefore support independent verification of the aggregate analyses, but not regeneration of trajectory-level detector flags from the original traces.

Ethics Statement

The benchmark experiments use public tasks and no private end-user data. The construct-validity study was conducted by two project-internal researchers, with disagreements adjudicated by a third project-internal researcher. No external participants were recruited, and the study involved no personal or sensitive data. This internal annotation activity did not require ethics review. The study evaluates the detector rather than the annotators, and the paper reports only aggregate labels. Human judgments supplied the annotations; automated tools were used to organize and analyze them, not to replace the annotators.

Agent trajectories in real deployments can contain source code, credentials, personal information, or behavioral signals. Applying the proposed evaluation procedure outside the public benchmark setting would therefore require project-specific privacy, access-control, retention, and security safeguards. The present results do not justify autonomous deployment of self-modifying components: proposed changes remain subject to sandboxing, attribution, reversibility, and independent validation, with human authors responsible for released artifacts and claims.

AI Use Statement

Generative AI systems were used for literature discovery and triage, research ideation, implementation and debugging, experiment orchestration, analysis scripts, and drafting or language editing of parts of the manuscript. The authors defined the research questions and analysis plan, checked cited sources, supervised the experiments, and verified quantitative claims against the underlying records. Human annotations were produced by the human annotators and adjudicator described in the ethics statement, not by an AI system. AI-generated text or suggestions were not treated as empirical evidence, no AI system is listed as an author, and the authors take full responsibility for the paper.

References

Elron Bandel, Asaf Yehudai, Lilach Eden, Yehoshua Sagron, Yotam Perlitz, Elad Venezian, Natalia Razinkov, Natan Ergas, Shlomit Shachor Ifergan, Segev Shlomov, et al. General agent evaluation. *arXiv preprint arXiv:2602.22953*, 2026.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4082.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. Sia: Self improving ai with harness & weight updates. *arXiv preprint arXiv:2605.27276*, 2026.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026a.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026b.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. Harnessbank: Semantic gene-bank search with gated verification for agent-harness self-evolution. *arXiv preprint arXiv:2607.13683*, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Atsuyuki Miyai, Kiyoharu Aizawa, and Toshihiko Yamasaki. Task-coevolve: Efficient harness optimization via adaptive validation task selection. *arXiv preprint arXiv:2608.20169*, 2026.

Xuying Ning, Dongqi Fu, Tianxin Wei, Hanqing Zeng, Yuanchen Bei, Bingxuan Li, Zihao Li, Qifan Wang, Xiang Shen, Yifan Wu, et al. Evoharness-rl: Learning self-evolving runtime harness for long-horizon llm agents. *arXiv preprint arXiv:2608.05446*, 2026.

Wenbo Pan, Shujie Liu, Chin-Yew Lin, Jingying Zeng, Xianfeng Tang, Xiangyang Zhou, Yan Lu, and Xiaohua Jia. Evolving agents in the dark: Retrospective harness optimization via self-preference. *arXiv preprint arXiv:2606.05922*, 2026.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, et al. Autosaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

Jinghan Xu, Yikai Zhang, Aili Chen, Weiyuan Li, Jiaqing Liang, and Deqing Yang. Verify smarter, evolve further: Efficient harness evolution through behavior-aware verification. *arXiv preprint arXiv:2608.27311*, 2026.

Chenyang Yang, Xinran Zhao, Tongshuang Wu, and Christian Kästner. Better harnesses, smaller models: Building 90% cheaper agents via automated harness adaptation. *arXiv preprint arXiv:2607.08938*, 2026.

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*, 2026.

Chenyu Zhao, Shenglin Zhang, Wenwei Gu, Yongqian Sun, Dan Pei, Chetan Bansal, Saravan Rajmohan, and Minghua Ma. Agenttether: Graph-guided diagnosis and runtime intervention for reliable llm agent operation. *arXiv preprint arXiv:2607.06273*, 2026.

A Experimental Protocol

A.1 Controlled design

The controlled study compares a static harness with the same harness augmented by two runtime checks: a repeated-command warning and a verification gate. Both arms use the same base model, the same 89 Terminal-Bench 2.0 tasks, and three nominal configuration batches labeled 0.2, 0.5, and 0.8. Archived request payloads are unavailable, so the labels are used for matching and do not establish that decoding temperature differed. This design yields 267 matched task–batch cells per arm and 534 controlled runs in total. Each run uses a common 100K pre-call token threshold, 25-turn limit, 2K output-token limit per turn, and 8K characters of observation context. The token

threshold is evaluated before a model call, so realized totals may overshoot: 249/534 runs exceeded 100K and the maximum was 117,827 tokens. Mean token use is 76.1K for the static arm and 78.1K for the monitored arm.

A.2 Outcomes, estimands, and inference

Verified success is a verifier reward equal to 1; zero, missing, or invalid rewards count as failures. Let p_a denote the verified-success rate in arm a. The success estimand is $\Delta_{\text{succ}} = p_{\text{monitored}} - p_{\text{static}}$. For the operational endpoint, let I_a denote the macro-average incidence of the four detector flags defined in Section B. We report $\Delta_{\text{det}} = I_{\text{static}} - I_{\text{monitored}}$, so positive values favor the monitored arm for both estimands.

The three conditions for a given task are not treated as independent replicates. Primary uncertainty estimates therefore resample the 89 tasks as clusters. The four-rule composite is the designated operational endpoint. Its definitions were fixed before mode-level analysis, after aggregate verified success had been inspected, and neither the definitions nor the decision threshold were externally preregistered.

A.3 Departures from the initial analysis plan

Table 7 summarizes the scientifically material departures from the initial plan and their implications for interpretation.

Table 7: Departures from the initial analysis plan and their implications for the reported analysis.

Planned	Implemented	Implication
Detector timing: fix the operational endpoint before mode-level analysis	Definitions were finalized after aggregate verified success was inspected but before mode-level statistics were computed; they were not externally preregistered	The composite is a designated operational endpoint, not a preregistered semantic-failure measure
Replicates: three independent seeds	Three nominal configuration batches, with one run per task–batch cell; request-level decoding parameters are unavailable	Inference clusters the 267 cells by 89 tasks, does not attribute batch differences to temperature, and does not estimate seed-level variability
Instrument calibration: apply calibrated command extraction and completion detection	Calibration preceded the analyzed data; records affected by earlier instrumentation faults were excluded and rerun	Only records collected under the calibrated measurement procedure enter the reported analysis
Model: use the initially specified model in both controlled arms	Both arms were moved to the same replacement model before its outcomes were inspected; earlier runs are retained only as exploratory context	The controlled estimand applies to the replacement model, not to a cross-model comparison
Invalid runs: obtain one valid observation for every planned cell	Reruns were restricted to documented instrumentation or execution failures	The task set, estimand, reward semantics, and verifier were unchanged by the replacements

B Operational Measures and Runtime Checks

B.1 Operational event definitions

A deterministic detector maps each trajectory to four binary flags. These flags are reproducible rules over recorded trace events, not validated measurements of semantic failure. The composite also mixes trace-derived rules with a terminal-outcome-derived rule.

Working-set overload. A run is flagged when at least one model reply is truncated and either no command is parsed or no action succeeds. The event count is the number of truncated replies when this conjunction holds.

Command-surface churn. Within an action frame defined by the command verb and first path operand, a run is flagged when a successful action is followed by a write-target change across paths sharing a basename or by a literal change. Explicit copy, move, or installation chains linking the paths are excluded, and actions already classified as retries are removed first. The initial analysis plan termed this rule "representation drift"; we use "command-surface churn" because that name matches the construct supported by the validation study below.

Uncontrolled retry. A run is flagged when the same whitespace-normalized command key fails at least twice; each failure from the second onward contributes one event. A terminal fold adds one event for timeout, budget exhaustion, or zero executable actions.

Premature completion. A run is flagged when it terminates with finished status but does not achieve verified success. A missing reward is not a success. This rule directly uses the verifier outcome and is therefore not outcome-independent.

Reply-, action-, and terminal-level events are computed separately. Retry classification precedes the churn rule, and task-level flags may co-occur. For mode m, let $d_m(\tau_i) \in \{0, 1\}$ indicate whether classifiable trajectory τ_i is flagged. Its arm-level incidence is $\mathrm{task_rate}_a(m) = N_a^{-1} \sum_{i=1}^{N_a} d_m(\tau_i)$, and the composite detector incidence is the macro-average of the four mode incidences. Event rates are reported separately. Infrastructure failures and incompatible trace schemas are excluded from detector denominators; the two controlled arms contain no such records.

B.2 Runtime checks

The monitored arm adds two checks to the otherwise unchanged harness.

Repeated-command warning. When the same normalized command fails twice with a diagnostic, the harness appends a warning asking the agent to diagnose the failure rather than repeat the command.

Verification gate. When the agent first declares completion early enough for a subsequent turn, the harness requests an objective verification command and then allows the run to continue. This check targets premature completion, whereas the repeated- command warning targets retry behavior. Neither check directly targets working-set overload or command-surface churn.

C Construct-Validity Study

C.1 Sampling and annotation protocol

The sampling frame comprises all 534 controlled task–arm–condition cells. Cells were stratified by arm, verified-success status, and detector-flag count (0, 1, or ≥ 2). The procedure allocated approximately equal numbers to nonempty strata and assigned twice the within-stratum sampling weight to cells flagged for command-surface churn. A fixed deterministic sampling procedure then drew with replacement, producing 100 annotation instances corresponding to 85 unique cells.

For each sampled instance, the annotation view hid the experimental arm, decoding condition, verifier reward, and detector labels. Two annotators independently labeled each operational mode using a fixed manual and the label set $\{0, 1, \mathrm{u}\}$, where u denotes insufficient evidence to make a binary judgment. A third adjudicator resolved all 19 segment–rule disagreements. Because sampling was with replacement, the study comprises 100 annotated instances rather than 100 independent controlled cells.

C.2 Agreement and detector correspondence

Table 8 compares the deterministic rules with the pre-adjudication union labels. Inter-annotator Cohen's κ was at most 0.49, and positive human examples were too sparse to support stable recall estimates.

Table 8: Human study of the operational trace-event rules on 100 sampled instances representing 85 unique controlled cells. Precision, recall, and F1 use the pre-adjudication union labels. They are unweighted sample-level diagnostics under the stratified design, not population-level detector performance estimates.

Rule	Rule positives	Human positives (A/B)	κ	P/R/F1
Overload	2	13/0	0.00	0.50/0.08/0.13
Churn	48	4/0	0.00	0.08/1.00/0.15
Retry	33	3/1	0.49	0.03/0.33/0.06
Premature	9	0/0	n.d.	0.00/n.d./n.d.

After adjudication, the human-positive counts were three for overload, one for churn, one for retry, and zero for premature completion. The single resolved churn positive was inside the 48 detector-positive instances, giving resolved precision 0.02 and recall 1.00 from one positive example. All resolved overload and retry positives were outside their detector-positive sets. These counts support the operational-detector interpretation used throughout the paper; they do not identify population-level semantic performance for the four rules.

D Additional Controlled Results

D.1 Success by condition and descriptive references

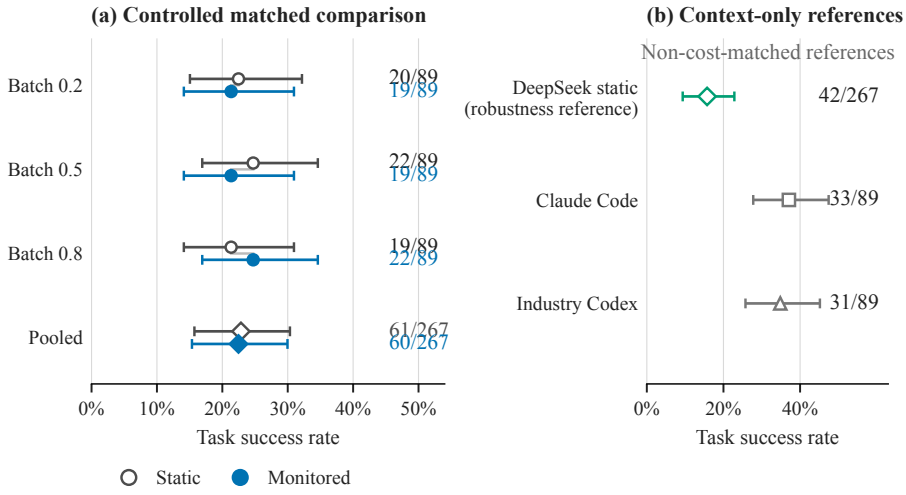

Figure 4: Verified success across three nominal configuration batches. Labels 0.2, 0.5, and 0.8 are archived batch metadata, not verified request-level decoding temperatures. Each batch contains the same 89 tasks; pooled estimates contain 267 runs per arm. Error bars are Wilson 95% confidence intervals for individual conditions and task-cluster bootstrap 95% confidence intervals for pooled estimates.

D.2 Intervention uptake

The exposure funnel differs across the two checks. The repeated-command warning fired in 6 of 267 monitored runs, whereas the verification gate fired in 63. The gate could act on completion declarations arriving within its implemented 15-turn eligibility horizon; 30 additional declarations arrived outside that gate horizon, although most were still within the 25-turn run limit. The checks target retry and premature completion, while overload and churn remain outside their target set; churn is the largest contributor to the composite endpoint.

Table 9: Controlled outcomes for the static and monitored arms, with descriptive model and harness references. DeepSeek and industry rows provide unmatched context; operational incidence is unavailable for the industry traces.

Arm or reference	Success	Rate	Avg. tokens	Detector incidence
Static	61/267	22.8%	76.1K	0.298
Monitored	60/267	22.5%	78.1K	0.295
Static (deepseek-v4-pro)	42/267	15.7%	89.6K	0.331
Industry (Claude Code)	33/89	37.1%	n/a	n/a
Industry (Codex)	31/89	34.8%	n/a	n/a

Table 10: Uptake of the two runtime checks in the 267 monitored runs.

Check	Exposure	Post-trigger behavior	Outcome
Repeat warning	177 eligible; 6/267 triggered (2.2%)	No post-warning retry event in 6/6; next command changed in 1/6	2/6 verified successes; retry events $6 \rightarrow 0$
Verification gate	93 declarations; 63/267 gated (23.6%); 30 late	57/63 issued a further command; 112 post-gate commands total	33/63 verified successes; 30/63 failed verification; 27/63 retained the premature flag

Among the 90 monitored runs ending with finished status, 43 were not verified successes: 42 had reward zero and one had a missing reward. Three gated runs without a premature-completion flag instead ended through budget exhaustion. This separation between terminal status and verifier outcome motivates reporting gate exposure and verified success as distinct evidence levels.

E Robustness and Sensitivity

The analyses in this section were conducted after the primary results were viewed and are descriptive. They were motivated in part by the construct-validity findings above. All confidence intervals resample tasks as clusters.

E.1 Mode-set and count-threshold sensitivity

Across the alternative detector sets in Table 11, estimated effects remain close to zero. Requiring at least two events per run gives $+0.4$ pp ($[-1.1, +1.9]$), and requiring at least three gives $+0.5$ pp ($[-0.8, +1.7]$); both intervals exclude the planned 5.96 pp reduction.

Table 11: Detector-set sensitivity. Effects are static minus monitored in percentage points with task-cluster bootstrap 95% confidence intervals.

Detector set	Static	Monitored	Effect [95% CI]
Four-rule macro	0.298	0.295	$+0.3$ $[-2.1, +2.8]$
Excluding churn	0.236	0.221	$+1.5$ $[-1.1, +4.4]$
Retry and premature only	0.335	0.320	$+1.5$ $[-1.9, +5.1]$

E.2 Triggered subsets

Table 12 compares monitored cells in which a check fired with their matched static cells. Triggering is post-treatment, so these comparisons are not causal estimates.

Table 12: Descriptive comparisons on triggered cells. Differences are static minus monitored in percentage points; intervals cluster by task.

Subset	Cells (tasks)	Wins M/S	Success diff. [CI]	Macro diff. [CI]
Repeat warning	6 (4)	2/2	-12.5 $[-75.0, +37.5]$	-9.4 $[-18.8, 0.0]$
Verification gate	63 (28)	33/30	-7.7 $[-20.8, +4.2]$	-1.9 $[-8.0, +4.6]$

E.3 Rule-threshold sensitivity

Raising one rule's event threshold at a time leaves all estimated effects near zero (Table 13). Retry counts combine repeated failures with the terminal fold, while premature-completion counts never exceed one.

Table 13: One-rule-at-a-time threshold sensitivity for macro detector incidence. Effects are static minus monitored in percentage points.

Variant	Effect [95% CI]	p
Overload ≥ 2 events	-0.2 $[-2.3, +2.1]$	0.84
Overload ≥ 3 events	-0.1 $[-2.3, +2.2]$	0.90
Churn ≥ 2 events	$+1.1$ $[-1.1, +3.5]$	0.37
Churn ≥ 3 events	$+1.4$ $[-0.8, +3.7]$	0.21
Retry ≥ 2 events	-0.4 $[-2.4, +1.7]$	0.71
Retry ≥ 3 events	-0.7 $[-2.6, +1.3]$	0.51
Premature ≥ 2 events	$+0.7$ $[-1.8, +3.2]$	0.61

E.4 Nominal-batch heterogeneity

No nominal batch drives the pooled result: each confidence interval includes zero, and the within-task heterogeneity tests are not significant (Table 14). Because request-level decoding parameters are unavailable, this analysis is a batch-stability check, not a temperature-effect analysis.

Table 14: Per-batch effects (static minus monitored, percentage points) and within-task heterogeneity tests. Batch labels are archived metadata and do not verify request-level temperature manipulation.

Endpoint	Batch 0.2	Batch 0.5	Batch 0.8	Heterog. p / consistent
Verified success	$+1.1$ $[-6.7, +10.1]$	$+3.4$ $[-4.5, +12.4]$	-3.4 $[-12.4, +4.5]$	0.54 / 83/89
Macro incidence	$+0.8$ $[-3.7, +5.3]$	$+2.0$ $[-2.5, +6.5]$	-2.0 $[-5.9, +2.0]$	0.45 / 65/89

F Persistent-Update Studies

These exploratory studies test two properties of persistent updating: end-to-end mechanism execution and transfer to later tasks. Their sample sizes support mechanism validation and bounded transfer observations rather than an efficacy estimate for the full evolution loop.

The scaled loop's churn-proxy incidence was 0.60 in the control and 0.73 in the evolve condition, a pattern consistent with verification and re-reading actions activating the command-churn proxy. Across these single-configuration studies, the update mechanism executes end to end, while persistent performance gains remain unestablished at the evaluated sample sizes.

G Coding Procedure for the Related-Work Matrix

Table 1 codes the cited manuscript versions available during the literature audit. For each of the 11 comparison systems, two passes over the main text recorded: (i) the implemented mechanism, (ii) whether evaluation used held-out or disjoint tasks, (iii) resource matching or accounting, (iv) a

Table 15: Design and outcomes of the exploratory persistent-update studies.

Study	Design	Observation
Mechanism study ($N = 3$)	Diagnose, propose rules, regression-gate, and deploy	Accepted rules were deployed; 0/3 tasks solved
Scaled loop ($N = 15$)	Same-order fixed-harness control	14 proposed edits, 2 accepted rule pairs; 2/15 evolve vs. 3/15 control; at least 52K regression-probe tokens
Disjoint transfer ($N = 15$)	Updated rule set vs. untouched harness on new tasks	4/15 vs. 4/15
Self-Harness re-implementation (Zhang et al., 2026)	30 edit tasks and 29 disjoint evaluation tasks	6/29 updated vs. 7/29 untouched

declaration that the relevant analysis was fixed before results, and (v) null, adverse, or regression-rejection outcomes. A checkmark requires an explicit implementation or report, a centered dot denotes partial coverage, and a cross means that coverage was not found in the audited main text. Supplementary materials and later manuscript versions were not independently re-coded.

The matrix is descriptive rather than a ranking. In particular, absence of a pre-specification declaration does not imply that all analyses in a work were post hoc, and held-out selection data are not automatically independent future-task evidence. For our row, the measurement component is an operational proxy with limited construct validation; resource control denotes shared nominal rules and reported consumption rather than exact equality; analysis was fixed internally without external preregistration; and persistent-transfer results remain exploratory.

H Reproducibility Details

The reproducibility materials contain the aggregate records and analysis code for the controlled comparisons. The paired exact tests and the success-rate and forest figures regenerate offline from the packaged tables and reproduce the reported values; trajectory-level analyses (four-mode classification, uptake funnel), the sensitivity and heterogeneity tables, and the exploratory persistent-update studies are provided as frozen aggregate records, because regenerating them would require the withheld raw trajectories. The controlled-run budgets, outcome definitions, and task-cluster analysis unit are specified above. The DeepSeek and industry references remain descriptive because their models, harnesses, and accounting are not matched to the controlled arms.

Operational detector classifications depend on the underlying command and event sequences. Raw trajectories are not distributed under the applicable benchmark, provider, and security constraints. The released aggregate records therefore support verification of the reported aggregate analyses, but not independent regeneration of trajectory-level detector flags.