CAN LLMS IMPROVE THEIR OWN HARNESS? A CONTROLLED STUDY ON ARC-AGI-3

Anonymous authors
Paper under double-blind review

ABSTRACT

A generally intelligent system acquires new skills efficiently without requiring humans to redesign its software scaffold for every new task, a longstanding aim in machine learning that ARC Prize and its community also build on. Language-model agents are a focus of attempts to meet this bar. Yet what actually acts on these benchmarks is rarely the model alone. It is a harness, a symbolic program that orchestrates calls to the model, handling perception, memory, tool use, recovery, and inference allocation on the model's behalf. ARC-AGI-3 is an interactive benchmark in which agents must explore unfamiliar environments, figure out their goals on the fly, and keep learning as they go. On it, human-steered systems already report ceiling scores across the public games, so we know effective designs exist. We ask whether frontier models can build and improve such a harness on their own. Through a fixed coding bootstrap, each model constructs a complete task harness, acts through that harness on five development games, and revises the entire codebase twice. Every version is evaluated on twenty games that never influence adaptation. We measure task performance with Relative Human Action Efficiency (RHAE), which combines level completion with action efficiency relative to human baselines, and record dollar-denominated inference cost as a practical resource measure. Across 27 lineages, harness performance ranges from zero to near-ceiling scores. Two revisions raise mean held-out RHAE by only 2.80 points (95% interval $[-1.44, 7.35]$). Development gains overstate held-out improvement, and reductions in inference cost can come at the expense of task performance. Human-steered systems demonstrate further headroom, although their development and selection procedures differ. These findings reveal a gap between autonomous harness construction and reliable adaptation under this two-revision protocol: even strong task performance does not ensure that subsequent revisions generalize or preserve existing capabilities.

1 INTRODUCTION

Modern language models can write programs, call tools, and act in unfamiliar environments, but only as part of a larger agentic system. In such a system, a symbolic program orchestrates calls to the model, shaping what it observes, what it retains, which actions are available, how it recovers from failure, and how much inference it spends. This surrounding system is commonly termed a harness (Lee et al., 2026; Lin et al., 2026a). A system's behavior therefore depends on both its harness and its underlying model or models.

This matters for the study of general intelligence, which requires the efficient acquisition of useful new skills in unfamiliar situations (Chollet, 2019). If each novel task requires a human engineer to inspect failures, redesign representations, add tools, and tune control flow, then what improves is not the system's own capacity to adapt but the engineer's understanding, transferred into the system by hand. The agent's learning remains confined to a supplied mechanism whose improvement depends on the engineer. In contrast, humans routinely adapt to unfamiliar situations in everyday life.

AI systems can improve through changes to their underlying models or to the harnesses through which those models act. These routes are complementary: harness changes can provide access to new tools, maintain persistent memory, and coordinate multiple model calls, capabilities that changes to model weights alone do not supply. The harness route therefore raises a central question:

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.

Can a model take on part of the engineering work that determines its own capabilities, using its execution experience to improve the software through which it acts?

ARC-AGI-3 aims to measure general intelligence through agents' adaptation to novel games. Its main metric, Relative Human Action Efficiency (RHAE), combines level completion with action efficiency relative to human baselines. Harnesses developed under human steering already reach 100 RHAE on the public games with GPT-5.6 Sol and Opus 5 (Lehmann et al., 2026; Han et al., 2026; Chen et al., 2026b). By contrast, Sol scores 38.3 on the public set in a simple benchmark harness with retained reasoning and compaction enabled (OpenAI, 2026). An off-the-shelf Claude Code baseline scores 42.83 with Opus 4.8/Fable 5 (Zeng et al., 2026). Strong human-steered results make ARC-AGI-3 a revealing test of autonomous improvement: effective harness designs are known to exist for current models. The benchmark lets us examine how reliably models can discover and improve their own designs through experience in unfamiliar interactive environments.

Recent systems automate agent and harness design by searching over prompts, skills, workflows, tools, memory, and source code (Zelikman et al., 2024; Hu et al., 2025; Zhang et al., 2025; Lee et al., 2026; Lin et al., 2026a; Zhang et al., 2026a). Our replicated study examines how reliably revisions generalize across model conditions, evaluating every prescribed harness snapshot on held-out games and measuring gains, regressions, and cost.

We ask a direct question: *Can a frontier reasoning model construct a complete interaction harness, learn from its execution record, and improve that same harness without human edits?* We measure the outcome of updating with a fixed-lineage protocol. Through a fixed coding bootstrap, a model starts from a controlled minimal repository, builds an ARC-AGI-3 harness for the same model profile, receives feedback from five development games, and potentially revises the generated harness twice. We then evaluate the initial, revision-1, and revision-2 snapshots on twenty holdout games.

Our results expose a gap between autonomous harness engineering and reliable self-improvement. Models autonomously produce diverse harnesses spanning zero to near-ceiling performance. Yet their revisions do not consistently turn development experience into held-out gains: mean improvement on development games exceeds held-out improvement by 5.62 RHAE points, and cost reductions can accompany performance losses. These findings show why the capacity to build an agent and the capacity to improve it need to be evaluated separately. We make three contributions:

- We provide a general framework for autonomous *harness adaptation* and define the harness adaptation gap: the difference between improvement on development tasks and improvement on held-out tasks.
- We introduce a replicated, all-snapshot protocol and conduct a controlled study across nine model–effort conditions: 27 independent harness lineages, 81 frozen snapshots, and 2,025 game evaluations, measuring task performance, deployment cost, and adaptation cost.
- We perform an outcome-blind, reliability-audited analysis of the observable capabilities in 54 revision episodes, connecting development feedback to code changes and relating those changes to performance on games withheld from revision.

2 Harness Adaptation as a Capability

Let a deployed agent be $S_t = (M, H_t)$, where M is a pretrained model and H_t is an executable harness. For a task τ, the inner loop maps an interaction history to actions under a fixed H_t. We write its expected utility over task distribution $\mathcal{D}$ as $J_{\mathcal{D}}(H_t; M)$, with M held fixed as the harness changes. The complete system is the unit of capability measurement (Russell & Subramanian, 1995; Kapoor et al., 2025). An outer-loop updater receives development evidence E_t^{dev} and produces

$$H_{t+1} = U(M, H_{\mathrm{boot}}, H_t, E_t^{\mathrm{dev}}).$$

The same model profile M appears in both loops. In the outer loop, it operates through a fixed general-purpose coding bootstrap H_{boot}, which supplies repository and development tools used to construct and revise H_t. In the inner loop, it operates through the task-specific harness H_t to interact with the environment. H_{boot} remains fixed while H_t is the object of autonomous adaptation. Meta-learning methods such as MAML, RL^2, and AdA learn rapid adaptation within a supplied neural architecture and training procedure (Finn et al., 2017; Duan et al., 2016; Bauer et al., 2023). Here, pretrained weights remain fixed while the model constructs and revises the executable task harness.

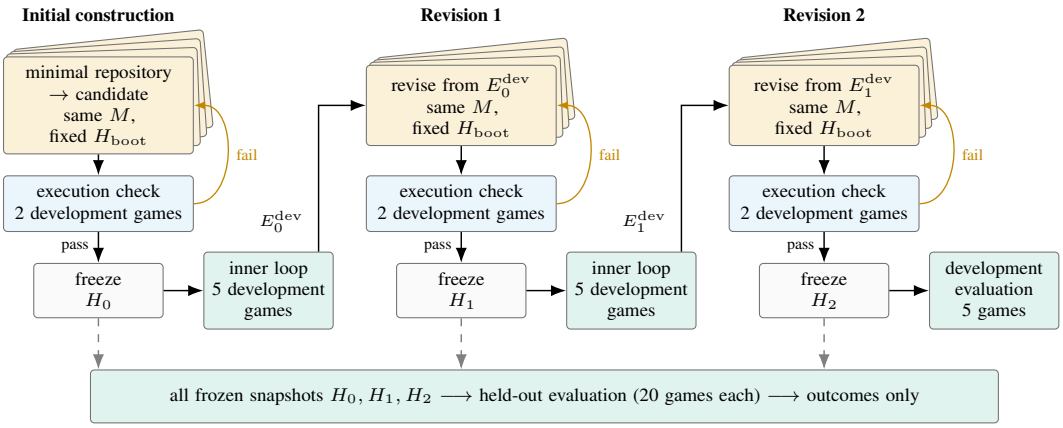

Figure 1: **Controlled harness-adaptation workflow.** At each stage, the same model profile uses the fixed coding bootstrap to propose a candidate. The controller checks that it can execute and interact with two development games. A failed candidate may be repaired; a passing candidate is frozen as H_t. Five-game development evidence from H_0 and H_1 seeds the next revision. H_2 runs on the same games for evaluation only. All snapshots are evaluated on twenty held-out games after adaptation.

We separate four capabilities: (i) constructing an executable harness, (ii) repairing interface and lifecycle failures, (iii) mapping execution evidence to a concrete update, and (iv) selecting updates that improve feedback-held-out tasks. The first three do not imply the fourth. A capable updater must also preserve working behavior and abstain when evidence is weak.

For revision t, define the harness adaptation gap

$$G_t = [J_{\mathcal{D}_{\text{dev}}}(H_t) - J_{\mathcal{D}_{\text{dev}}}(H_0)] - [J_{\mathcal{D}_{\text{test}}}(H_t) - J_{\mathcal{D}_{\text{test}}}(H_0)] .$$

A positive G_t means that apparent adaptation on feedback-generating tasks exceeds held-out benefit. Let $C_{\text{run}}(H_t)$ denote the expected execution cost per task and $C_{\text{outer}}(H_{0:t})$ the cumulative cost of constructing and revising the harness through stage t. We evaluate performance and both costs jointly:

$$(J_{\mathcal{D}_{\text{test}}}(H_t), -C_{\text{run}}(H_t), -C_{\text{outer}}(H_{0:t})) .$$

This follows bounded-optimal and resource-rational accounts in treating computation as a costly action (Russell & Wefald, 1991; Lieder & Griffiths, 2017; 2020). Our Pareto comparisons concern held-out performance and per-game deployment inference cost; construction and revision expenditure is reported separately.

In our ARC-AGI-3 instantiation, task utility is a game's RHAE score, and we estimate $J_{\mathcal{D}_{\text{dev}}}$ and $J_{\mathcal{D}_{\text{test}}}$ by mean RHAE over the five development and twenty held-out games, respectively. We record C_{run} and C_{outer} as dollar-denominated inference costs for gameplay and harness development.

3 Related Work

LLM self-refinement and program optimization. Self-Refine, Reflexion, and CRITIC revise responses or task policies through self-feedback, retained verbal feedback, or tool-grounded critique (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024). Prompt and program optimizers learn reusable components from development data. OPRO and PromptBreeder search instruction space. DSPy, MIPRO, TextGrad, and GEPA optimize modules inside declared language-model programs (Yang et al., 2024; Fernando et al., 2024; Khattab et al., 2024; Opsahl-Ong et al., 2024; Yuksekgonul et al., 2025; Agrawal et al., 2026). These methods normally retain a human-specified artifact schema or computation graph.

Automated agent and harness design. Prior work automates optimizer construction, code-defined agent design, and workflow search (Zelikman et al., 2024; Hu et al., 2025; Zhang et al., 2025), and extends autonomous revision to runtime logic and complete coding-agent repositories (Yin et al., 2025; Robeyns et al., 2025; Zhang et al., 2026a). Meta-Harness and Agentic Harness Engineering

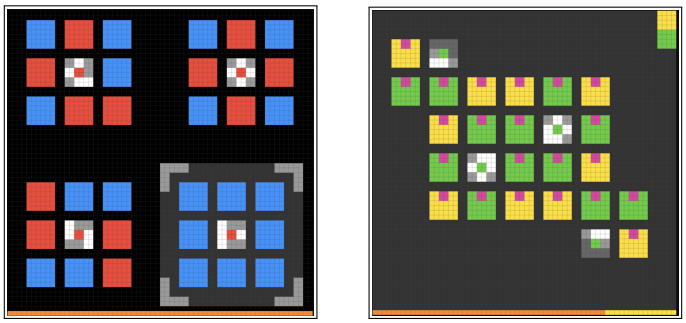

Figure 2: **Two observations from FT09 (Level 1 on the left, and Level 7 on the right), a public ARC-AGI-3 game.** The agent must infer that the small patterned tiles specify which neighboring tiles must match or differ from a reference color. Clicking cycles tile colors, which need to be matched to target patterns. The later levels introduce controls that change several tiles together. These rules must be discovered through interaction.

optimize executable harness source from scores and traces (Lee et al., 2026; Lin et al., 2026a). HarnessCompass and RRSI study how such changes generalize beyond the tasks used for evolution (Zhang et al., 2026b; Xia et al., 2026). We study autonomous harness engineering as a capability of frontier models: how reliably can they construct the software through which they act and improve it from their own execution evidence? Our replicated comparison follows each independently constructed harness and both prescribed revisions, measuring initial capability alongside subsequent gains and regressions on games excluded from adaptation.

Evaluating autonomous improvement. SEAGym reports a schedule that completes 39 of 40 harness updates while test success falls by 14.5 percentage points (Zheng et al., 2026). On Terminal-Bench, validation-selected evolution yields only a small mean held-out gain (Wang et al., 2026); HarnessOpt-Bench finds that the best validation score seen during search is usually optimistic relative to the nominated harness's test score (Ursekar et al., 2026). Our construction-and-revision study extends this evidence to independently generated interactive-game harnesses, jointly tracking performance on development and held-out games and inference cost.

Harness design for ARC-AGI-3. EWM, OPINE-World, PRO-LONG, and Tycho demonstrate strong autonomous game solving within human-steered harnesses (Rodionov, 2026; Courtis et al., 2026; Fox et al., 2026; Lehmann et al., 2026). Workspace Optimization lets agents revise executable world models and strategies during play while the reusable harness remains fixed (Sarafian et al., 2026). Our study shifts harness design and revision to the model: it constructs the reusable task harness and revises it from its own execution evidence, without human edits or guidance between stages. We evaluate whether those revisions improve performance on games excluded from adaptation. Appendix C compares the protocol designs.

4 Experimental Design

4.1 ARC-AGI-3: An Interactive Reasoning Challenge

Unlike the static input–output transformations in earlier ARC tasks, ARC-AGI-3 presents an agent with sequential, unfamiliar grid worlds. At each turn, the agent observes one or more 64×64 color grids, together with the current game state, progress across levels, and available actions. Depending on the game, the interface offers directional inputs, a game-specific input, coordinate clicks, or undo. The agent is not told what the visible objects represent, how these inputs affect them, or what completes a level. It must discover the game's rules through exploration and reasoning.

Each game contains multiple levels. Relative Human Action Efficiency (RHAE) assigns credit only to completed levels and discounts solutions that use more environment actions than hidden human baselines. Later levels receive greater weight. We also record actions, completed levels, terminal states, and dollar-denominated model inference cost.

Figure 2 shows two states from one such game, FT09 [1]. A harness can present these states to the runtime model as raw images, symbolic grids, object summaries, transition diffs, or combinations of them. It also controls context and memory, provides tools, organizes planning and reasoning, and allocates model calls. It can request one action or a sequence, use code to execute a discovered strategy, and reuse computations to reduce inference cost. These design choices are not specified by the benchmark.

4.2 AUTONOMOUS CONSTRUCTION AND REVISION

Two harnesses, one model. Each experimental condition pairs a model with a reasoning-effort setting and uses that profile for both harness development and gameplay. During construction and revision, the model operates through OpenCode, a general-purpose coding harness in the same category as Claude Code and Codex. OpenCode provides tools to inspect and edit files and run commands. It remains fixed throughout the campaign. During gameplay, the model is called through the generated task harness, the artifact being autonomously constructed and improved.

Each run receives a sandboxed workspace with Python, preinstalled scientific libraries, and shell tools. The repository includes documentation for interacting with the game, a client for calling the assigned model, and model and pricing specifications. The model must create `run.sh`, which launches its solver on a specified game without human input. The solver architecture is otherwise unspecified. The harness controls how much work is performed by deterministic code and when the model is consulted. The build model may create and revise solver code and run local checks, including mock-based tests.

Coding sessions do not expose the evaluator's live game or runtime-model gateways. End-to-end trials occur through controller-run smoke qualification on two declared development games. If a candidate fails, the same model receives its failure report and execution logs and may repair it for at most ten cycles. Qualification requires the harness to run and interact with the game within fixed action and level limits, regardless of its RHAE score. Task performance is measured in the subsequent fixed-snapshot evaluations.

The qualified initial system runs on five development games. Revision 1 receives a fixed bundle containing its own source, traces, scores, actions, and inference costs. It may change any generated file or preserve the parent. The same procedure produces revision 2. No human edits or natural-language guidance are introduced between stages. Resource limits and infrastructure-recovery rules are specified in Appendix A.

To assess the outcomes of construction and revision, we compare the resulting harness versions under a common evaluation protocol.

- **Held-out evaluation.** Every frozen harness version is evaluated on the five development games and twenty held-out public games. The held-out games provide no feedback for harness construction, revision, or selection. Our primary comparison is the change in mean held-out RHAE from the initial harness to revision 2 among runs completing all three stages. Construction failures are reported separately over all planned runs. We validate recorded gameplay by replaying the same actions locally and through the official ARC service, checking agreement in scores and game progress (Appendix G).
- **Conditions and uncertainty.** The study covers nine model–effort conditions across seven model families, with three independent constructions per condition (Table 1). All 27 runs complete construction and both revisions. All conditions follow the same construction-and-revision protocol. We estimate confidence intervals by bootstrapping whole lineages, preserving paired comparisons across versions. These intervals describe uncertainty for the observed study pool on the fixed game split. Comparisons between model–effort conditions are descriptive.

[1] `https://arcprize.org/tasks/ft09`

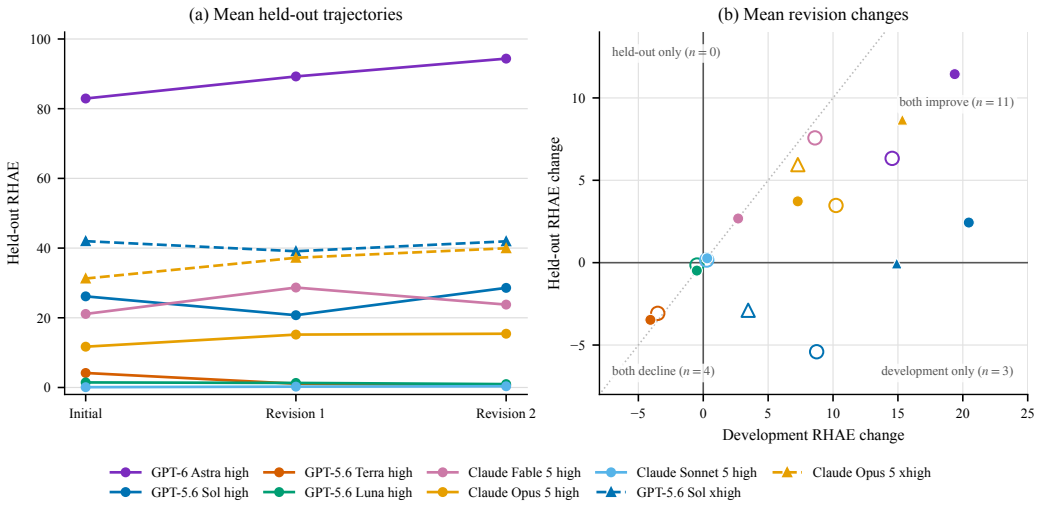

Figure 3: **Construction strength varies, and development gains transfer unreliably.** (a) Held-out condition means on a common scale. (b) Each condition has two points, averaging its three runs: initial-to-revision-1 (open) and initial-to-revision-2 (filled). Labels count strict-sign quadrants. Points below zero regress on held-out games. Triangles denote higher-effort conditions.

5 Quantitative Results

We measured execution variability in a dedicated repeated-evaluation study. Ten evaluations of one frozen final harness on the same twenty held-out games yield a sample SD of 3.99 RHAE. All ten scores remain above this lineage's initial-harness score, showing that its large observed improvement persists across repeated evaluations. The observed variation also shows why score changes of only a few points require cautious interpretation (Appendix D.1). Held-out gains vary substantially across lineages. Revision 2 raises mean development RHAE by 8.42 points and held-out RHAE by 2.80 points. The held-out 95% bootstrap interval is $[-1.44, 7.35]$, compatible with no average benefit or a useful positive gain. Observed strict-sign changes comprise 15 gains, 12 losses, and no ties ($p = .70$, two-sided sign test). At a descriptive five-point threshold, these counts become 9 gains, 3 losses, and 15 within-threshold outcomes (Appendix D). These are observed suite outcomes, not estimates of how often revisions reduce expected performance.

The pooled result also hides differences between games. Averaging across runs, held-out performance improves on 13 games and declines on seven. The largest mean gain is 20.44 RHAE points and the largest decline is 15.03. Aggregate improvement can therefore coexist with substantial regressions on particular games (Appendix D).

Development gains overstate transfer. The mean adaptation gap G_2 is 5.62 points (95% bootstrap interval $[1.78, 9.77]$). Figure 3b shows several revisions that improve the five feedback-generating games while reducing held-out performance. Development improvement therefore overstates the average held-out gain under this update procedure. A post hoc rule selecting the snapshot with highest development RHAE yields a mean held-out gain of 4.53 points over the initial harness (95% interval $[0.81, 8.68]$), compared with 2.80 for always deploying H_2. Mean deployment cost is \$9.15 rather than \$7.57 per game; the rule remains to be evaluated prospectively (Appendix D).

Half of the cheaper final harnesses also improve performance. For each immutable snapshot, solving cost is the mean logical inference cost over the 20 held-out games, including replayed responses. Across all 27 lineages with complete selected-game cost meters, nine revision-2 descendants improve both held-out RHAE and cost per game. Six gain RHAE at higher cost, nine become cheaper while losing RHAE, and three worsen on both dimensions. Mean cost falls by \$2.72 per game, but the median change is only $-\$0.54$. The lower average expenditure thus combines revisions that achieve more at lower cost and revisions that spend less while also solving less effectively.

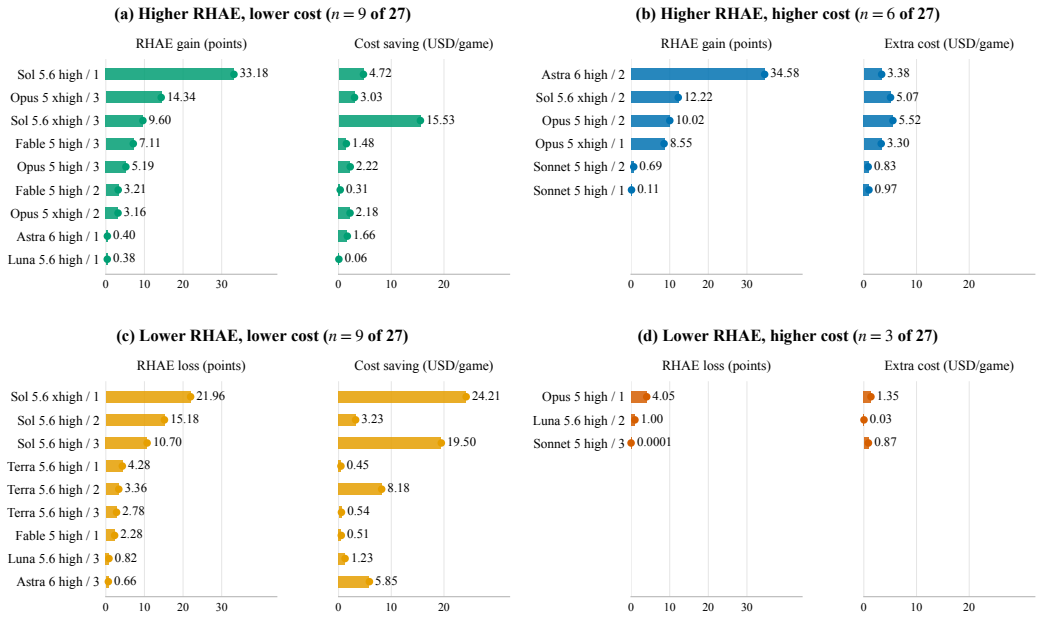

Figure 4: **Separate outcomes reveal which harnesses improve, save cost, or trade one for the other.** Each row identifies a model–effort condition and run. Bars show magnitudes of observed $H_2 - H_0$ changes in mean held-out RHAE and inference cost per game; headers specify the direction. Each metric uses a common linear scale across panels. Categories use unrounded changes.

The experimental program corresponds to at least $18,129.27 under the frozen publication price schedule, including construction, revision, repair, qualification, and fresh selected-game inference. Appendix D provides the cost breakdown, complete replicate-level trajectories (Table 3), and checkpoint and game-sensitivity analyses.

6 What Models Build and How They Revise

6.1 What harness designs emerge, and how do they change?

The generated systems span markedly different interaction strategies. Compact direct model controllers coexist with heuristic-first systems, strategist–executor hybrids, persistent manuals, explicit world models, replay systems, deterministic planners, and inference-budget managers. Architectures also vary across replicates of the same condition.

Across all 27 lineages, we audit all 54 parent-to-descendant transitions. This post hoc descriptive audit labels a revision as *Preserve* when it repairs or tunes the current policy, *Extend* when it adds a capability inside the same decision loop, *Reorient* when it materially changes control priority, context continuity, replay, or action/inference scheduling, and *Replace* only when it changes the harness family. These labels describe observable source and runtime structure; they are separate from the outcome-blind capability coding below.

Strategy change is common. Twenty-two of 27 first revisions reorient the harness, four extend it, and one primarily preserves or restores the incumbent. Second revisions include 12 reorientations, 12 extensions, and three preserves. No lineage replaces its harness family; revisions repeatedly reorganize memory, perception, batching, replay, and resource allocation. Appendix E reports every replicate and Figure 5 summarizes size and code churn.

One Sol-high revision addressed decision-limit failures and ineffective memory use by changing action budgeting, memory management, and observations. Held-out RHAE rose from 29.24 to 53.56 at essentially unchanged inference cost (Appendix E). In a Sol-xhigh lineage, revisions replaced

response chaining with independent calls and bounded memory. Held-out RHAE fell from 43.95 to 21.99 as inference cost fell from $31.39 to $7.18 per game (Appendix E).

6.2 Which revision behaviors accompany held-out improvement?

We examine how models respond to execution feedback and whether their revision behaviors accompany held-out gains.

Corpus and codebook. The coding corpus contains all 54 parent-to-descendant revision episodes across the nine conditions, comprising 70 updater sessions. Packets preserve the development feedback available to the updater, compact chronological messages and tool events, the exact selected-system diff, and proximal test and smoke results. They mask model and condition identity, trajectory identifiers, future stages, and every downstream development and held-out outcome. The codebook covers evidence use, planned and enacted changes, verification, and post-result adaptation (Appendix F).

Blinding and reliability. Each outcome-blind packet was independently coded twice using GPT-5.6 Sol xhigh under the same frozen codebook. Consensus labels were frozen before linkage to downstream outcomes. We report unadjudicated exact agreement, quadratic-weighted Cohen's κ (Cohen, 1968), and Krippendorff's α (Krippendorff, 2004). Scope ($\kappa = 0.96$), post-result adaptation (0.84), and planning (0.70) pass the exploratory reliability threshold. Updater verification is uniformly top-coded with exact agreement, so its chance-corrected reliability is undefined. The planning interval is absent from 27 packets, with missingness concentrated in several conditions and larger edits; we therefore exclude it from outcome interpretation. Diagnostic probing (0.66), diagnosis (0.35), incumbent protection (0.47), and resource metareasoning (0.29) fall below the preset 0.67 reliability threshold. Visible narration is triangulated against tools, code, tests, and runtime behavior because verbalized reasoning may be unfaithful (Turpin et al., 2023; Lanham et al., 2023; Chen et al., 2025).

Do broader edits accompany larger gains? All 54 revision episodes include inspection of item-specific development evidence, all final diffs target at least one eligible evidence-backed issue, and every selected descendant passes its proximal operational checks. Completing an evidence-guided revision does not ensure a better agent: all episodes satisfy these process checks, while held-out outcomes vary. Broad rewrites average $+3.75$ immediate held-out change versus -0.20 for contained changes, with a wide interval that includes zero. Appendix F reports the complete exploratory association table.

How do reactive revision episodes differ? In 36 episodes, the updater links the final result to its change and hands off. Their mean immediate held-out change is $+2.48$. In 18 episodes, a post-edit result triggers another edit, revert, test change, scope change, or criterion-based stop. Their mean is -0.76. The -3.24-point contrast is uncertain: its interval includes zero. Episodes also differ in parent performance, repair effort, and the changes attempted. The comparison therefore establishes neither a dependable advantage nor a harmful causal effect of further repair (Appendix F).

7 Discussion

Construction strength does not ensure consistent improvement. Across nine conditions, the study establishes substantial autonomous harness-building capability while exposing a gap between development gains and held-out benefit. The pooled interval remains compatible with useful average improvement, and several lineages achieve large gains. Reliability here concerns consistent held-out benefit and capability preservation; observed regressions do not establish each revision's effect on expected performance. Evidence inspection, targeted edits, and operational validation are common, but their exploratory associations cannot establish why revisions succeed or fail. Strong initial harnesses can lose observed performance while weaker constructions improve substantially.

A gap to human-steered harnesses remains. At the same Opus 5 xhigh setting, Tycho reaches 100 RHAE on every public game, whereas our revision-2 harnesses average 39.95 on twenty of those games (Lehmann et al., 2026). Both studies designate the same five development games, although

Tycho also uses the full public set for policy selection. Tycho reports 100 with Sol, while VISTA and AVO report 100 with Opus 5 (Han et al., 2026; Chen et al., 2026b). Retrodict reports 99.86 with Sol at maximum effort (Brown, 2026). These results establish substantial headroom for autonomous harness engineering. The development and selection procedures differ, so they do not isolate an effect of human steering. Appendix D gives the comparison scope.

Abstention is part of adaptation. Although retaining the parent is permitted, every scored trajectory changes its implementation. Reliable self-improvement includes recognizing when the evidence favors keeping the current harness. The post hoc development-based selection result illustrates why generating a revision and deciding whether to use it are distinct parts of autonomous improvement. Candidate revisions should be compared with a preserved baseline and rejected or rolled back when they do not justify replacement. RRSI's reported gains with edit budgets and noise- and cost-aware acceptance rules illustrate one way to make such decisions explicit (Xia et al., 2026).

Implications for autonomous improvement. Harness adaptation complements model training by changing the software through which fixed model weights act (Lee et al., 2026; Lin et al., 2026a). For AI risk analysis, current task performance is only part of the picture: a system may also be able to improve the software that determines its future performance (Wijk et al., 2025). Model and harness improvement could reinforce each other if better harnesses support AI research and better models become more capable harness developers (Chen et al., 2026a). Evaluating this route requires tracking whether revisions generalize, preserve prior strengths, and justify their cost. Our experiment holds both the model and coding bootstrap fixed. Claims about sustained capability acceleration or its risks require longer runs, broader tasks, and updaters that can themselves change (Wijk et al., 2025; Chen et al., 2026a).

8 Limitations

We use one benchmark, one set of feedback games, and two revisions. The results therefore leave open how harness adaptation behaves on other tasks or over longer improvement sequences. Because the games are public, some models may have encountered them during training.

Three runs per condition leave uncertainty about small differences between models. Gameplay can also vary even with an unchanged harness, as our ten-run diagnostic shows. We did not repeat every evaluation, so individual gains or regressions may partly reflect this variation.

Inference expenditure varies across harnesses and models. Reporting score and cost together makes this visible, but does not separate model capability, harness design, and spending into independent effects.

The revision-behavior annotations in Section 6.2 were produced by two LLM annotation passes. Their agreement with independent human annotations was not evaluated. The coders use summarized execution records, which can lose context or event order. We exclude unreliable or inadequately observed labels. Episodes differ in their starting harness and the problems they attempt to fix, so an association cannot show that a particular revision behavior caused the outcome.

9 Conclusion

Harness adaptation offers a route to self-improving AI in which a system develops the software through which it acts. Our ARC-AGI-3 study demonstrates substantial autonomous harness-building capability, but a less dependable ability to convert execution experience into held-out gains. Development progress can overstate improvement, and half of the cheaper final harnesses lose held-out performance. Reported human-steered systems show that stronger harnesses are possible for some of the same models. Strong performance from a model–harness system should be distinguished from its ability to autonomously improve the software through which it acts. Our results make this distinction concrete: with model weights fixed, development gains and lower inference costs can coexist with held-out regressions, making transfer and preservation of existing capabilities essential tests of harness adaptation. The central challenge is to make improvements accumulate: extending what the system can do across tasks while preserving what it already does well.

AI USE STATEMENT

Language-model assistants supported experimental-infrastructure implementation, code and methodological review, literature search, interpretation checks, manuscript editing, and the outcome-blind qualitative coding described above. The authors had the initial idea, selected the protocol and data, verified the analyses and claims, authored the final text, and take responsibility for the work.

REPRODUCIBILITY STATEMENT

Appendices A and B specify the protocol and starting prompt; Appendix C compares related systems. Appendices D–F provide supplementary results, strategy audits, and the coding method. The review artifact contains the cleaned framework, frozen prompts and splits, all selected source snapshots, analysis tables, deterministic reproduction scripts, and a checksum-backed manifest (Appendix G).

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M. Zhang. Human-timescale adaptation in an open-ended task space. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1887–1935, 2023. URL https://proceedings.mlr.press/v202/bauer23a.html.

Ryan Brown. Retrodict: An agent for ARC-AGI-3. Software repository and experiment report, 2026. URL https://github.com/ryanbbrown/Retrodict.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in AI: From bounded self-refinement to autonomous research loops. *arXiv preprint arXiv:2607.07663*, 2026a. URL https://arxiv.org/abs/2607.07663.

Terry Chen, Yeyin Zhu, Zhifan Ye, Jean-François Puget, and Humphrey Shi. NVIDIA AVO reaches 100% on ARC-AGI-3, demonstrating a frontier-level general-purpose architecture for long-horizon autonomous agents. NVIDIA Technical Blog, 2026b. URL https://developer.nvidia.com/blog/nvidia-avo-reaches-100-on-arc-agi-3-demonstrating-a-frontier-level-general-purpose-architecture-for-long-horizon-autonomous-agents/.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025. URL https://arxiv.org/abs/2505.05410.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019. URL https://arxiv.org/abs/1911.01547.

Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220, 1968. doi: 10.1037/h0026256.

David Courtis, Wenhao Li, and Scott Sanner. OPINE-World: Programmatic world modeling with ontology-error-prioritized interactive exploration for ARC-AGI-3. *arXiv preprint arXiv:2607.01531*, 2026.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779*, 2016. URL https://arxiv.org/abs/1611.02779.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Alexis Fox, Junlin Wang, Paul Rosu, and Bhuwan Dhingra. PRO-LONG: Programmatic memory enables long-horizon reasoning. *arXiv preprint arXiv:2607.20064*, 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, pp. 57734–57811, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Qiushi Han, Keya Hu, Linlu Qiu, Cathy Wu, and Kaiming He. VISTA: A visual harness for reasoning in an interactive world. Project release, 2026. URL https://vista-research.github.io/.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=Zy4uFzMviZ.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f1cf02ce09757f57c3b93c0db83181e0-Abstract-Conference.html.

Klaus Krippendorff. Reliability in content analysis: Some common misconceptions and recommendations. *Human Communication Research*, 30(3):411–433, 2004. doi: 10.1111/j.1468-2958.2004.tb00738.x.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

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*, 2026. URL https://arxiv.org/abs/2603.28052.

Jens Lehmann, Andrei Aioanei, and Sahar Vahdati. Tycho: Active abstraction with programmatic world models for ARC-AGI-3. *arXiv preprint arXiv:2607.28287*, 2026. URL https://arxiv.org/abs/2607.28287.

Falk Lieder and Thomas L. Griffiths. Strategy selection as rational metareasoning. *Psychological Review*, 124(6):762–794, 2017. doi: 10.1037/rev0000075.

Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. doi: 10.1017/S0140525X1900061X.

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*, 2026a. URL https://arxiv.org/abs/2604.25850.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving LLM agents. *arXiv preprint arXiv:2605.30621*, 2026b. URL https://arxiv.org/abs/2605.30621.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019.

OpenAI. How enabling two settings tripled our scores on the ARC-AGI-3 benchmark. Official research report, 2026. URL https://openai.com/index/how-two-settings-tripled-our-arc-agi-3-scores/. Accessed 2026-09-25.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.525.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025. URL https://arxiv.org/abs/2504.15228.

Sergey Rodionov. Executable world models for ARC-AGI-3 in the era of coding agents. In *Artificial General Intelligence*, volume 16855 of *Lecture Notes in Computer Science*, pp. 198–210. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-032-33195-3_15.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991. doi: 10.1016/0004-3702(91)90015-C.

Stuart J. Russell and Devika Subramanian. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research*, 2:575–609, 1995. doi: 10.1613/jair.133.

Elad Sarafian, Gal Kaplun, Ron Banner, Daniel Soudry, and Boris Ginsburg. Workspace optimization: How to train your agent. *arXiv preprint arXiv:2605.09650*, 2026. URL https://arxiv.org/abs/2605.09650.

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, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377.

Alexy Skoutnev, Kirill Acharya, Gaston Longhitano, Madeleine Udell, Kevin Ellis, and Iddo Drori. Twin: Playing an unknown game with a test-time digital twin. *arXiv preprint arXiv:2608.14490*, 2026. URL https://arxiv.org/abs/2608.14490.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965. Curran Associates, Inc., 2023. doi: 10.52202/075280-3275.

Varun Ursekar, Apaar Shanker, Yash Maurya, Shehab Yasser, Vijay S. Kalmath, Veronica Chatrath, and Yuan Xue. Harnessopt-bench: Evaluating LLMs at harness optimization. *arXiv preprint arXiv:2608.06301*, 2026. URL https://arxiv.org/abs/2608.06301.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents. *arXiv preprint arXiv:2607.12227*, 2026. URL https://arxiv.org/abs/2607.12227.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M. Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Peng Xia, Rujun Han, Zifeng Wang, Yanfei Chen, Yufan Zhuang, Yoonho Lee, Chengsong Huang, Han Yu, Zhongying CuiZhu, Yifei Ming, Huaxiu Yao, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. RRSI: Regularized recursive self-improvement of agent harnesses. *arXiv preprint arXiv:2609.24972*, 2026. URL https://arxiv.org/abs/2609.24972.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, pp. 12028–12068, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1354.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU.

Guanning Zeng, Jiani Wang, Wenjie Ma, Shaofeng Yin, Chenyang Wang, Shichen Liu, Angjoo Kanazawa, Wode Ni, Xiuyu Li, Andrea Zanette, and Haiwen Feng. [schema]: Frontier models with the right harness achieve approximately 99% on ARC-AGI-3 public. Impossible Research project release, 2026. URL https://schema-harness.github.io/.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/aa5f5e6eb6f613ec412f1d948dfa21a5-Abstract-Conference.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, Yizhou Jin, and Yan Xu. Harnesscompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses. *arXiv preprint arXiv:2608.01918*, 2026b. URL https://arxiv.org/abs/2608.01918.

Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. SEAGym: An evaluation environment for self-evolving LLM agents. *arXiv preprint arXiv:2606.17546*, 2026. URL https://arxiv.org/abs/2606.17546.

A Full Experimental Protocol

The nine model–effort conditions each have three independent constructions and follow the same construction-and-revision protocol (Table 1). All conditions enter the pooled analysis.

Repository boundary. The initial repository provides the task contract, local engine client, controlled model client, pricing documentation, and the entry-point contract. It does not provide a solver architecture, game-specific rules, worked examples, or a reusable task policy. Generated systems have no internet access, no cross-game state, and no access to other game instances. The fixed coding bootstrap is OpenCode; the artifact records its binary and adapter fingerprints. Codex is used separately for outcome-blind annotation.

Development split. Five development games were selected at random before the campaign: `tr87`, `vc33`, `r11l`, `bp35`, and `ft09`. The other 20 public games were fixed as feedback-held-out. The two smoke games were chosen from the development split to exercise click- and keyboard-based interaction.

Stage lifecycle. Each stage has one selected candidate. Controller-owned smoke qualification checks interface validity and bounded interaction on two declared development games. Each game permits 50 charged actions and stops after two completed levels or a win. Passage requires a charged action, a clean exit, and a registered stopping boundary on each game; no score threshold is used. Failed candidates may be repaired by the same condition for at most ten cycles. The selected initial candidate runs the five-game development evaluation. Its frozen source and feedback bundle seed revision 1. The same procedure seeds revision 2. All attempts, including failed repair and report-finalization sessions, remain in the audit ledger.

Hosted replay. Generated systems interact only with the local engine. The controller freezes complete action–observation traces, replays them locally, then submits them to a short-lived hosted scorecard. Acceptance requires score agreement and matching compared observations, with one explicitly logged visual-noise exception described in Appendix G. Hosted service behavior therefore cannot influence the generated policy.

Construction replacement policy. A construction may be replaced only for controller, provider-transport, or hosted service failures that occur outside model-produced behavior. Syntax errors, invalid protocol use, exhausted repair budgets, and failure to produce a qualified harness remain model or harness outcomes. Every planned construction in the retained study cohort produced a qualified initial system.

Table 1: Study conditions. Builder and runtime profiles are matched within each row. Context is the advertised window; max output is the configured per-response gameplay cap. Coding caps are given below. Exact deployment and pricing-profile hashes are retained in run protocols.

Model profile	Reasoning effort	Context	Max output
GPT-6 Astra	high	1.05M	64K
GPT-5.6 Sol	high	1.05M	120K
GPT-5.6 Terra	high	1.05M	120K
GPT-5.6 Luna	high	1.05M	120K
Claude Fable 5	high	1.00M	120K
Claude Opus 5	high	1.00M	120K
Claude Sonnet 5	high	1.00M	120K
GPT-5.6 Sol	xhigh	1.05M	120K
Claude Opus 5	xhigh	1.00M	120K

Resource bounds. Coding and gameplay sessions each have a 24-hour wall-clock limit. A generated workspace or selected snapshot may occupy at most 1 GiB. Model calls use a 3,600-second retry budget and the output limits in Table 1. Coding output caps are 128K for GPT-5.6 and Claude profiles and 64K for GPT-6 Astra. Exact storage limits, model profiles, and recovery controls are included in the artifact. Gameplay inference has no common total-dollar or model-call cap: expenditure is measured, and the build prompt asks systems to use inference economically, while the stated wall-clock, action, output, and storage bounds remain enforced.

RHAE definition. For a game with levels $\ell = 1, \ldots, L$, hidden human action baselines b_ℓ, and system actions a_ℓ, a completed level within the $5b_\ell$ action budget receives $s_\ell = \min\{100(b_\ell/a_\ell)^2, 115\}$ and other levels receive zero. With level weights $w_\ell = \ell$, the game score is

$$\min\left\{\frac{\sum_{\ell=1}^{L} w_\ell s_\ell}{\sum_{\ell=1}^{L} w_\ell}, 100\frac{\sum_{\ell=1}^{C} w_\ell}{\sum_{\ell=1}^{L} w_\ell}\right\},$$

where C is the completed prefix. Reported split scores average this quantity over games. The second term caps credit by weighted completion.

B Minimal Starting Prompt

The build model receives the following instruction. The controller substitutes only the runtime model's display name and configured reasoning effort. In particular, the prompt does not prescribe a solver architecture, visual representation, memory design, planning method, or rule for any game.

> **Build a system for solving ARC-AGI-3 games. Work autonomously to implement, test, and document the complete solution without asking for human input.**
>
> The system will be evaluated on the tradeoff between RHAE and inference cost. Aim primarily for high RHAE while using inference economically. Use additional inference only when it is reasonably expected to produce a meaningful improvement in RHAE. Inference cost is measured in US dollars according to `docs/pricing.md`. RHAE rewards action efficiency. When outcomes have equal RHAE, fewer game actions are preferred.
>
> At evaluation, the system receives exactly one current game and must infer its mechanics from observations available during that run. It runs with Python 3.12 without internet access or access to other games. It may use the configured runtime model only through the supplied `llm_client.py`. Exact model capabilities and limits are in `docs/model.md`.
>
> The local rules and interface are in `docs/arc_rules.md`. The supplied files are read-only. Expose a non-interactive `./run.sh --game <game-id> --out-dir <directory>` entrypoint.

C Related-System Comparison

Table 2 compares the evaluation unit, selection rule, and editable artifact. Our protocol evaluates each individual descendant of a frozen parent on held-out tasks that did not influence the update, with no performance-based selection among qualified descendants. HUIB separately evaluates models' ability to produce useful updates and to use updated artifacts in an online task stream (Lin et al., 2026b); here, one model profile fills both roles.

Table 2: Closest protocol precedents. Strict test means that test tasks do not influence proposals, acceptance, stopping, or candidate selection.

Work	Editable artifact and schedule	Final evidence	Main protocol difference
STOP	One optimizer function over 1–4 recursive depths	Fresh optimization tasks	Restricted to an optimizer-function artifact.
ADAS	Code-defined agent functions under archive-guided search	All discovered ARC agents on held-out test; five evaluations per agent	Searches agent functions; our study tracks prescribed revisions of complete interactive harnesses.
AFlow / GEPA	Workflows or prompts under budgeted search	Validation-selected candidate on test	Constrained artifact and selection over many candidates.
SICA / DGM	Complete coding-agent codebase with archive-guided evolution	Same-benchmark selection plus transfer	Whole-code evolution with variable lineages and update counts.
Meta-Harness / AHE	Executable harness source under multi-iteration score-and-trace search	Best or final frozen harness	Seeded candidate search with selected-harness outcomes.
RRSI (Xia et al., 2026)	Harness source; 20–40 rounds with regularized proposals and acceptance	Evolved harness on held-out tasks and OOD benchmarks	Tests a selection intervention under multi-candidate search, rather than every prescribed descendant.
HarnessCompass	Minimal Bash-only coding harness over five rounds	Final harness on 450 untouched tasks	Close minimal-seed strict test in a different domain.
HUIB / SEAGym / Rethinking	Skills, prompts, memory, or seeded harnesses	In-stream or strict-test evaluation	Studies artifact use, regressions, and allocation of rollout budgets.
HarnessOpt-Bench	Executable seed under trusted execution	Inaccessible strict test after one nomination	Candidate search and final nomination introduce selection.
Workspace Optimization	Structured ARC workspace with online repair within games	Public 25-game suite	Artifact repair occurs within individual games.
Ours	**Complete ARC harness with exactly two revisions**	**Every snapshot on 5 development and 20 strict held-out games**	**Prescribed lineages; primary endpoint is the final revision. Joint RHAE, actions, and dollar cost.**

D Additional Quantitative Results

Table 3: All completed trajectories reported in the paper. Conditions are grouped by provider, with matched higher-effort profiles first. Logical costs use the publication price schedule and selected per-game token ledgers; three trajectories are recovered directly from those ledgers because final aggregate report assembly failed.

OpenAI

Condition	Rep.	RHAE / cost (USD/game) Initial	Revision 1	Revision 2
GPT-6 Astra high	1	97.14/11.90	93.77/10.86	97.55/10.24
	2	58.03/8.58	80.76/12.53	92.62/11.96
	3	93.59/18.77	93.23/11.87	92.94/12.92
GPT-5.6 Sol xhigh	1	43.95/31.39	18.49/4.89	21.99/7.18
	2	37.64/12.62	56.37/13.09	49.86/17.69
	3	44.36/37.89	42.43/23.73	53.96/22.36
GPT-5.6 Sol high	1	20.38/12.87	29.24/8.19	53.56/8.15
	2	27.92/8.84	6.82/2.90	12.74/5.61
	3	30.15/25.39	26.17/6.26	19.45/5.89
GPT-5.6 Terra high	1	4.77/4.68	0.94/4.70	0.49/4.24
	2	4.39/9.88	1.72/1.55	1.04/1.70
	3	3.33/3.89	0.60/2.49	0.55/3.34
GPT-5.6 Luna high	1	1.52/0.46	1.71/0.57	1.90/0.40
	2	1.75/0.14	1.87/0.44	0.75/0.17
	3	1.09/2.00	0.33/0.85	0.26/0.77

Anthropic

Condition	Rep.	RHAE / cost (USD/game) Initial	Revision 1	Revision 2
Claude Opus 5 xhigh	1	27.85/9.18	28.79/8.18	36.39/12.48
	2	43.66/13.62	46.01/12.76	46.82/11.44
	3	22.29/10.77	36.84/8.65	36.63/7.75
Claude Opus 5 high	1	6.57/1.12	3.79/2.71	2.52/2.46
	2	16.46/3.93	21.47/7.16	26.48/9.45
	3	12.09/6.78	20.28/4.30	17.28/4.56
Claude Fable 5 high	1	16.05/10.63	18.37/11.26	13.77/10.12
	2	28.35/17.29	41.41/15.99	31.55/16.98
	3	18.92/13.81	26.26/14.15	26.04/12.33
Claude Sonnet 5 high	1	0.00/0.27	0.00/0.33	0.11/1.24
	2	0.12/0.88	0.21/1.48	0.81/1.70
	3	0.14/0.49	0.52/1.24	0.14/1.36

Table 3 gives the complete held-out score–cost trajectory for each completed construction, with full model versions and reasoning-effort profiles.

Construction and qualification outcomes. All 27 planned runs in the retained study cohort produce a smoke-qualified initial system and complete all three evaluations. Paired revision analyses use all 27 complete lineages. Across their 81 scored stages, 46 qualify the raw candidate without repair and 35 select a smoke-repaired candidate. Initial held-out RHAE averages 24.54 and ranges from 0 to 97.14.

Uncertainty estimation. Percentile confidence intervals use 100,000 whole-lineage bootstrap resamples with seed 20260902 and the artifact's fixed pseudonymous trajectory order. Each resampled lineage retains all stages and games, preserving paired comparisons across versions. Intervals condition on the observed study pool and fixed game split, not new tasks or feedback partitions.

Cost accounting. Logical solving cost sums priced usage for fresh and replayed responses in each suite-selected game attempt; the per-game meters remain auditable independently of final experiment-report assembly. For cross-model comparison, dollar values are recomputed from the immutable token ledgers using first-party standard API list prices frozen on September 13, 2026; regional and marketplace premiums are excluded. Original run-time pricing profiles and hashes remain in the artifact. These dollar figures cover model inference; they exclude infrastructure and human labor and do not measure latency or energy use.

Table 4: Descriptive held-out means for matched reasoning-effort conditions. Each row contains three independent trajectories.

Condition	Initial	Revision 2	Change
Sol high	26.15	28.58	+2.43
Sol xhigh	41.98	41.94	-0.05
Opus high	11.71	15.43	+3.72
Opus xhigh	31.27	39.95	+8.68

Development-based checkpoint selection. As a post hoc diagnostic, selecting the preserved snapshot with highest five-game development RHAE chooses $H_0/H_1/H_2$ in 9/6/12 lineages and yields 29.06 mean held-out RHAE: $+4.53$ over H_0 (95% interval $[0.81, 8.68]$) and $+1.72$ over always deploying H_2 ($[0.38, 3.67]$). Mean deployment cost rises from \$7.57 to \$9.15 per game. The rule uses only development scores and could be automated; its post hoc evaluation does not establish the performance of a prospectively specified selection policy (Table 5).

Table 5: Post hoc checkpoint diagnostic. Development selection chooses the snapshot with highest five-game development RHAE, breaking ties toward the earlier snapshot. "$+/-/=$" counts held-out improvement, regression, and no change relative to H_0. Costs exclude outer-loop construction and selection expenditure. The held-out oracle is diagnostic only.

Deployment policy	RHAE	Gain	USD/game	$+/-/=$	$H_0/H_1/H_2$
Retain H_0	24.54	0.00	10.30	0/0/27	27/0/0
Deploy H_2	27.34	+2.80	7.57	15/12/0	0/0/27
Select by development RHAE	29.06	+4.53	9.15	13/5/9	9/6/12
Held-out oracle	30.66	+6.13	9.80	18/0/9	9/8/10

Table 6: Logical solving inference cost per held-out game across all 27 scored lineages. Logical cost includes replayed responses.

Snapshot	Mean USD/game	Median USD/game
Initial	10.30	9.18
Revision 1	7.15	6.26
Revision 2	7.57	7.18

Table 7: Joint held-out score–cost movements. The "cheaper/RHAE down" category records joint decreases in cost and RHAE.

Transition	Pareto gain	Score↑/cost↑	Cheaper/RHAE↓	Worse both	Tie/cost↓	Tie/cost↑
$H_0 \to H_1$	6	9	9	2	0	1
$H_1 \to H_2$	6	6	8	7	0	0
$H_0 \to H_2$	9	6	9	3	0	0

Table 8: Inference cost in USD under the frozen publication price schedule. Selected-game cost counts fresh model inference; logical replay charges used for score–cost comparison are excluded. Coding-bootstrap cost covers construction, revision, and repair; qualification is runtime inference used to select a smoke-qualified candidate.

Component	Initial	Revision 1	Revision 2	Total
Coding loop, 27 scored	607.92	371.78	372.64	1,352.34
Smoke qualification, 27 planned	408.02	111.11	127.72	646.84
Fresh selected-game attempts, 27 scored	6,749.95	4,712.60	4,667.55	16,130.09
List-price-equivalent total	**7,765.89**	**5,195.49**	**5,167.90**	**18,129.27**

The experimental program spanned at least five weeks across two machines and corresponds to at least $18,129.27 under the frozen publication price schedule. Fresh selected-game attempts account for $16,130.09. Across the 27 scored lineages, coding-bootstrap inference for construction, revision, and repair totals at least $1,352.34. Smoke qualification adds $646.84 across all planned runs.

We report endpoint score and solving cost jointly: nine descendants are strict deployment score–cost Pareto gains, while nine apparent savings are performance contractions. Where aggregate report accounting is unavailable, logical cost is recovered from immutable suite selections and complete per-game meters. Unselected game-attempt recovery overhead is outside the reported program lower bound (Appendix G). Coding cost contains three unresolved requests, all in revision 1; qualification contains seven. We do not impute these requests, so Table 8 reports lower bounds. The same treatment applies to one further unresolved provider request outside these two cost components.

Human-steered reference systems. Human steering refers to development decisions and is compatible with AI-written code. Tycho reports 100.00 RHAE over all 25 public games with GPT-5.6 Sol at maximum reasoning effort and with Opus 5 at xhigh (Lehmann et al., 2026). VISTA and AVO also report 100.00 with Opus 5 (Han et al., 2026; Chen et al., 2026b); VISTA uses xhigh and reports 98.27 for its separate Sol-max run. Retrodict reports 99.86 with Sol at maximum effort (Brown, 2026). In our study, revision-2 held-out means are 28.58/41.94 for Sol at high/xhigh and 15.43/39.95 for Opus at high/xhigh (Table 4). The Opus comparison matches both model version and reasoning effort. Tycho designates the same five development games as our study (`tr87`, `vc33`, `r11l`, `bp35`, `ft09`) and scores 100 on each of the twenty remaining games. The performance gap therefore persists on the same evaluation games. The Sol comparison uses different reported effort labels (max versus high/xhigh).

The development and selection procedures differ. Tycho selects among four policies using Opus 4.8 on all 25 public games, then evaluates the selected policy with Opus 5 and Sol; our twenty held-out games provide no feedback for construction, revision, or selection. Retrodict likewise reports earlier development attempts on all public games. Reported costs vary substantially across these systems and use different pricing dates and accounting conventions; we do not normalize their historical costs to our frozen publication price schedule. The comparison establishes attainable performance with these models, but does not isolate the contribution of human steering or establish equivalent efficiency.

The coding-agent baselines cited in the introduction come from within-study comparisons. Schema reports 42.83 RHAE for its generic Claude Code baseline and 98.98 for its world-model harness using the same Opus 4.8/Fable 5 pairing (Zeng et al., 2026). Twin reports 61.1 for off-the-shelf Codex with Sol and 93.3 with its world-model harness, holding the model, file bridge, and sandbox fixed; the stronger harness also uses more inference compute (Skoutnev et al., 2026).

Table 9: Endpoint robustness to practical RHAE thresholds. Changes within $\pm\epsilon$ are indeterminate; deployment Pareto gains also require lower solving cost.

ϵ	Improve	Regress	Indeterminate	Pareto gain
0	15	12	0	9
1	11	8	8	7
5	9	3	15	5

D.1 FIXED-HARNESS EXECUTION VARIANCE

The study evaluates each frozen snapshot once. To measure run-to-run execution variance separately from construction and revision, we reran one snapshot after the campaign: Sol-high replicate 1 at H_2. The snapshot was selected post hoc after campaign outcomes were available, and GPT-5.6 was chosen for its operational speed; this is a diagnostic case rather than a random sample of harnesses. The original evaluation ran in August 2026 and the nine fresh evaluations in September 2026. Recorded harness, controller, model-profile, provider-deployment, and environment fingerprints are identical across evaluations, although unexposed provider-side changes cannot be excluded. Each evaluation uses the same twenty held-out games and starts new runtime-model inference. No construction, revision, repair, checkpoint selection, or hosted replay occurs in the added runs.

Table 10: Ten evaluations of one unchanged Sol-high H_2 harness. Evaluation 0 is the original campaign observation. "Complete" counts games for which all levels were solved; RHAE also gives partial credit within the remaining games.

Evaluation	RHAE	Actions	Levels completed	Complete games
0	53.56	8,492	102	8
1	46.86	8,509	94	6
2	43.38	9,382	90	6
3	48.21	9,389	92	7
4	45.32	8,515	91	6
5	55.37	8,459	104	8
6	46.52	9,435	99	7
7	49.23	9,410	92	6
8	52.10	8,648	96	8
9	44.75	10,135	93	5

Across all ten evaluations, mean RHAE is 48.53, the sample standard deviation is 3.99, and the range is 43.38–55.37. These are descriptive statistics conditional on the post hoc snapshot. The nine fresh evaluations alone have mean 47.97 and sample standard deviation 3.79, so the observed variability does not depend on including the older campaign observation. The original 53.56 observation is the second highest and lies 5.03 points above the ten-evaluation mean. The mean absolute difference between two evaluations is 4.73 points. Every evaluation remains above this trajectory's 20.38 initial score, so its endpoint improvement classification is unchanged. Replacing only its original H_2 observation by the ten-run mean changes the campaign-wide endpoint mean from $+2.80$ to $+2.62$; this is a mechanical sensitivity check rather than a measurement-error correction for the other trajectories.

Variation is larger at game level: the median within-game sample standard deviation is 18.78 RHAE points, and the three largest are 30.76 on `sb26`, 28.33 on `m0r0`, and 27.13 on `tn36`. Averaging twenty games therefore substantially stabilizes the suite score, while single-trajectory differences of a few points remain difficult to interpret precisely. The observed 3.99-point suite standard deviation gives empirical scale to the $\epsilon = 5$ sensitivity in Table 9; one harness is insufficient to define a universal noise threshold. These post hoc evaluations are separate from the construction-and-revision study, and their spend is excluded from Table 8. The supplement releases all 200 evaluation–game rows and a standalone reproduction command for these statistics.

Weighting and game sensitivity. The endpoint mean is $+2.80$ under both trajectory and condition weighting, since each condition has three complete lineages, and $+2.55$ when the seven model families receive equal weight. The condition-stratified bootstrap interval is $[-1.12, 7.01]$ under either weighting and remains inconclusive. Leaving out one condition at a time yields means from $+1.72$ to $+3.59$. Across held-out games, 13 have a positive mean endpoint change and seven a negative one; the largest gain is $+20.44$ on `lp85`, while the largest decline is -15.03 on `sb26`. The small pooled gain is distributed across conditions and concentrated in opposing game-level effects. These intervals condition on the fixed games and do not quantify uncertainty over new development partitions.

Omitting `r11l` only from the observed development summary reduces the mean development–held-out gap from 5.62 to 3.02 points. The corresponding gaps after omitting each of the other development

games range from 4.68 to 7.84. This is a descriptive influence check, not a counterfactual experiment: all harnesses were still revised using the original five-game feedback. Tables 11 and 12 give completion, action, and per-game context. Higher level completion accompanies the pooled score gain, but aggregate action counts alone do not distinguish improved efficiency from earlier stopping. The supplement reproduces these summaries from the same published game rows.

Table 11: Held-out completion and action context across the same 540 evaluations per stage (27 lineages $\times$ 20 games). Level completion is the mean within-game fraction of completed levels. Actions are mean charged actions; lower action counts need not imply better efficiency because systems may stop without completing a game. These summaries do not isolate the components of RHAE.

Stage	RHAE	Complete games	Levels completed	Actions
Initial	24.54	73/540 (13.5%)	34.5%	259.6
Revision 1	25.87	83/540 (15.4%)	36.4%	292.0
Revision 2	27.34	94/540 (17.4%)	37.1%	273.6

Table 12: Per-game initial-to-revision-2 changes, averaging the same 27 paired lineages. Completion columns are percentage-point changes in complete games and mean fraction of levels completed. Rows describe observed outcomes; no alternative feedback split was used to generate the harnesses.

Game	Split	ΔRHAE	ΔComplete	ΔLevels	ΔActions
bp35	dev	+3.43	+3.7	+5.3	+46.5
ft09	dev	-0.45	+11.1	-2.5	-2.1
r11l	dev	+18.83	+14.8	+17.9	+22.1
tr87	dev	+8.11	+7.4	+11.1	+22.8
vc33	dev	+12.19	+11.1	+13.2	+76.1
ar25	heldout	-1.29	+3.7	-4.2	-80.7
cd82	heldout	+3.11	+3.7	+1.9	-1.0
cn04	heldout	+1.74	+3.7	+0.0	-39.9
dc22	heldout	-2.39	+3.7	-4.3	+21.2
g50t	heldout	-1.42	+0.0	-0.5	-4.5
ka59	heldout	-1.46	+7.4	-1.6	-13.0
lf52	heldout	-3.75	-3.7	-4.4	+41.1
lp85	heldout	+20.44	+29.6	+13.0	-24.1
ls20	heldout	+4.96	+3.7	+7.4	+87.0
m0r0	heldout	-1.49	-3.7	+0.0	-49.6
re86	heldout	+1.38	+3.7	-0.5	-3.0
s5i5	heldout	+8.25	+7.4	+8.3	-33.6
sb26	heldout	-15.03	-18.5	-13.0	+5.0
sc25	heldout	+1.64	+0.0	+2.5	+46.6
sk48	heldout	+4.69	+3.7	+5.1	+72.4
sp80	heldout	+10.01	+11.1	+11.7	+74.6
su15	heldout	+4.82	+0.0	+7.4	+37.8
tn36	heldout	+4.49	+7.4	+5.3	+44.1
tu93	heldout	+15.66	+14.8	+16.9	+59.7
wa30	heldout	+1.71	+0.0	+1.2	+39.9

E Harness and Strategy Audit

We inspect every selected snapshot, exact diff, updater record, and relevant smoke-repair artifact in all 27 trajectories. The strategy labels are post hoc descriptions derived with outcome visibility. *Preserve* (P) means repair or tuning without changing the operating strategy. *Extend* (E) adds a material capability within the same decision loop. *Reorient* (R) changes control priority, context continuity, replay, or action/inference scheduling while retaining the broader harness family. *Replace* would change that family; no transition does so.

Every lineage grows from the initial to the second-revision snapshot. Median size rises from 2,161 to 2,604 to 2,912 generated Python lines, and the mean endpoint increase is 1,052 lines. Both large and

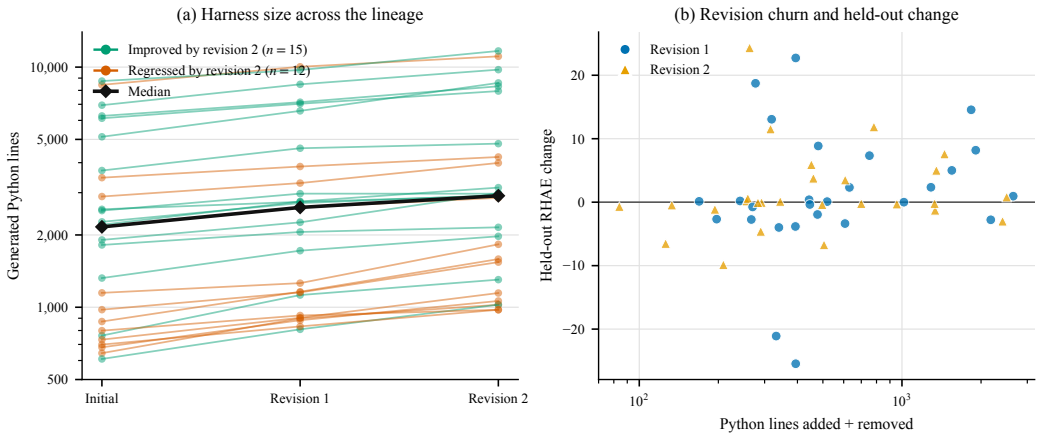

Figure 5: **Harnesses grow and undergo substantial revision without reliable gains.** Generated Python counts model-produced source and tests and excludes the supplied `llm_client.py`. (a) All 27 lineages on a log scale, colored by initial-to-revision-2 held-out outcome; the black path is the stage median. (b) Each point is one selected parent-to-descendant Python diff. Churn is added plus removed lines, and points above zero improve held-out RHAE.

small diffs lie on each side of zero held-out change. Edit volume establishes substantive revision; held-out change determines whether that revision improved performance.

Table 13: Strategy-transition counts across all conditions. Entries give Preserve/Extend/Reorient.

Condition	Initial harness family	$H_0 \to H_1$ P/E/R	$H_1 \to H_2$ P/E/R
Astra high	Persistent Python workers with policy reuse	0/0/3	0/1/2
Sol high	Direct multimodal batched planner	0/0/3	0/3/0
Terra high	Compact visual/text planner	0/0/3	0/2/1
Luna high	Direct model controller	0/2/1	0/2/1
Fable high	Strategist, executor, and replay system	1/0/2	1/0/2
Opus high	Layered world-model/planner hybrids	0/0/3	0/0/3
Sonnet high	Heuristic/model hybrid	0/1/2	1/2/0
Sol xhigh	Multimodal batched planner	0/1/2	1/2/0
Opus xhigh	World-model and replay hybrids	0/0/3	0/0/3
All conditions		1/4/22	3/12/12

First revisions reorganize; second revisions diversify. First revisions most often change the operating strategy itself: models alter context persistence, replay flow, model-consultation cadence, action batching, or budget allocation. Second revisions split equally between extending capabilities and reorienting the strategy. Even changes described as incremental by their builders can change when the runtime model is consulted or how scarce actions and dollars are scheduled.

Replicates disagree within the same condition. The strategy paths are identical across all Sol-high replicates but their held-out outcomes range from a 33.18-point gain to a 15.18-point loss. Fable, Luna, Sonnet, and Sol-xhigh also mix preserve, extend, and reorient transitions within one condition. Opus-high and Opus-xhigh reorient at every stage, yet their outcomes include both gains and regressions. Strategy change is therefore common, and its presence or coarse type does not certify improvement.

An evidence-linked improvement. Sol-high replicate 1 (T001), selected post hoc for illustration, connects feedback to a substantial gain. Three development games exhausted H_1's self-imposed 100-decision limit; its meters recorded repeated cache writes with few reads. Revision 2 raised this internal limit to 120, removed explicit cache writes, and added animation-aware observations, object

Table 14: Strategy transitions for all 27 replicates. P, E, and R follow the definitions above; Table 3 gives the corresponding score–cost trajectories.

Condition	Replicate	$H_0 \to H_1$	$H_1 \to H_2$
Astra high	1	R	R
Astra high	2	R	R
Astra high	3	R	E
Sol high	1	R	E
Sol high	2	R	E
Sol high	3	R	E
Terra high	1	R	E
Terra high	2	R	E
Terra high	3	R	R
Luna high	1	E	E
Luna high	2	E	R
Luna high	3	R	E
Fable high	1	R	P
Fable high	2	P	R
Fable high	3	R	R
Opus high	1	R	R
Opus high	2	R	R
Opus high	3	R	R
Sonnet high	1	R	E
Sonnet high	2	E	P
Sonnet high	3	R	E
Sol xhigh	1	R	E
Sol xhigh	2	E	P
Sol xhigh	3	R	E
Opus xhigh	1	R	R
Opus xhigh	2	R	R
Opus xhigh	3	R	R

geometry, and completed-level memory. The pre-edit plan names these changes; the transcript records 17 passing local tests, and the candidate qualified without repair. Held-out RHAE rose from 29.24 to 53.56 while inference cost per game changed from \$8.19 to \$8.15. Several components changed together, so the case illustrates evidence-grounded adaptation with observed transfer without isolating a causal mechanism. The supplement's `case-study/` directory links feedback excerpts, the plan, source, tests, and outcomes.

Cost reduction with performance loss. In Sol-xhigh replicate 1 (T007), the first revision replaced response-ID chaining and history compaction with independent calls carrying a bounded notebook and recent outcome. Across the two revisions, held-out RHAE fell from 43.95 to 21.99 and inference cost from \$31.39 to \$7.18 per game; development RHAE also fell, from 37.11 to 31.73. Revision 2 partially recovered both scores. Several components changed together, so any attribution of the regression to lost contextual information remains untested. The supplement's `case-study/T007.md` links the source changes and per-stage outcomes.

F Capability Codebook and Sampling

The process-coding corpus covers all 54 parent-to-descendant revision episodes from the 27 study lineages. Each packet receives two independent annotations under the same frozen codebook, with model identity and downstream outcomes masked.

To relate observable revision behavior to held-out outcomes, we analyze visible model messages, tool calls and results, generated source, exact diffs, tests, fixed feedback packets, smoke records, runtime traces, actions, scores, and costs. We do not treat provider-hidden chain-of-thought as evidence of cognition.

Evidence boundary and units. Direct trace evidence is a logged event or immutable artifact. Operational relations are reproducible links, such as a feedback item followed by a targeted edit and relevant test. Claims about latent beliefs or understanding are interpretations and are excluded from quantitative coding. Complete trajectories are the independent units; events, games, and stages are nested observations.

Codebook organization. The codebook measures four stages of the update process:

1. feedback-evidence access, diagnostic specificity, and discriminating probes.
2. plan enactment, feedback coverage, edit alignment, and change scope.
3. incumbent protection, updater verification, and proximal result.
4. post-result adaptation and resource allocation.

Table 15: Episode codebook used for observable harness revision.

Code group	Central question	Observable indicators
Evidence access	Does the updater inspect item-specific support before editing?	Feedback summaries, traces, logs, score breakdowns, errors, and reproducers (EVID).
Diagnosis	Does it connect cited evidence to a mechanism and prediction?	Pre-edit symptoms, localized mechanisms, predictions, counterfactuals, and boundary conditions (DIAG).
Diagnostic probing	Does evidence discriminate among hypotheses?	Targeted inspection, stated alternatives, refutable expectations, and result-governed narrowing (PROBE).
Plan and alignment	Are targets, actions, criteria, and retained edits linked?	Enacted plans, eligible-feedback coverage, and final-edit alignment (PLAN, COV, ALIGN).
Change control	Does it contain risk and protect the incumbent?	Executable scope, fallbacks, baseline checks, comparison, narrowing, rollback, and no-change decisions (SCOPE, PROTECT).
Verification	Does it test the retained change and remain runnable?	Reproducers, targeted and regression tests, smoke repair, requalification, and proximal results (VERIFY, RESULT).
Post-result adaptation	Does a post-edit result govern the next action?	Interpretation followed by handoff, edit, revert, changed test, narrowed scope, or criterion-based stop (ADAPT).
Resource adaptation	Does evidence govern computation allocation?	Calls, context, batching, caching, tests, stopping, numeric triggers, and expected-value comparisons (META).

Codebook refinement and consensus. An initial blind calibration exposed temporal ambiguities. We tightened only the pre-edit and post-result anchors; every packet was then coded under frozen version 2 before its outcome linkage was opened. Consensus labels were frozen by a conservative rule: exact agreement stands, ordinal disagreement receives the lower evidenced capability level, and any missing interval remains missing.

Ordinal episode coding. The twelve variables are feedback-evidence access (EVID), diagnostic specificity (DIAG), discriminating probe strength (PROBE), plan enactment (PLAN), eligible-feedback coverage (COV), final-edit alignment (ALIGN), scope containment (SCOPE), incumbent protection (PROTECT), updater verification (VERIFY), proximal operational result (RESULT), post-edit result-grounded adaptation (ADAPT), and resource adaptation (META). Sequential codes credit only evidence visible before the relevant action. Final summaries can corroborate but cannot create a prior diagnosis, prediction, probe, plan, or resource criterion. Mandatory final smoke passage contributes to RESULT, but raises VERIFY only when a failure causes repair and requalification.

Table 16: Unadjudicated reliability under frozen codebook version 2. n excludes variable-specific missingness. κ_w uses quadratic disagreement weights $d_{ij} = ((i-j)/(K-1))^2$ for K ordered categories. α uses nominal disagreement and exposes prevalence effects. Dashes denote undefined chance-corrected coefficients for invariant labels.

Code	n	Exact	κ_w	α
EVID	54	1.00	–	–
DIAG	30	.67	.35	.38
PROBE	53	.85	.66	.67
PLAN	27	.85	.70	.70
COV	54	1.00	–	–
ALIGN	54	1.00	–	–
SCOPE	54	.93	.96	.88
PROTECT	54	.74	.47	.47
VERIFY	54	1.00	–	–
RESULT	54	1.00	–	–
ADAPT	54	.93	.84	.84
META	35	.71	.29	.17

Table 17: Exploratory process associations for labels with adequate inter-session reliability and outcome coverage. Contrasts are immediate held-out RHAE change at the highest minus lowest shown code. Intervals resample complete trajectories; p values permute trajectory profiles. Associations are descriptive.

Code	n	κ_w	Contrast/counts	Interpretation
Scope containment	54	.96	-3.94	Scoped minus broad, interval $[-9.09, 1.22]$, $p = .12$.
Post-result adaptation	54	.84	-3.24	Further-action minus handoff, interval $[-8.54, 2.38]$, $p = .174$.

Outcome linkage. Coder files and conservative consensus labels were hashed before opening the private episode key. Associations use immediate parent-to-child held-out change. Confidence intervals resample trajectories. Permutation tests shuffle complete two-stage outcome profiles across trajectories while preserving stage. Our exploratory inclusion rule requires $\kappa_w \geq .67$; codes below this cutoff and invariant high-agreement codes are reported descriptively. The artifact provides aggregate reliability, prevalence, outcome, and association tables and fixes 10,000 resamples and permutations per process association with deterministic seeds. These tests are exploratory and unadjusted for multiple comparisons. Shuffling across conditions assumes exchangeable trajectory profiles, which condition-level differences may violate; the reported p values do not remove confounding.

Planning-label missingness. The planning interval is observable in 27 packets and absent in 27. Missing packets average 9.22 executable files changed and 1,147 lines added, compared with 3.74 files and 414 lines when planning is observed. Missingness is also condition-dependent: all Opus-high, Opus-xhigh, and Sonnet packets are missing, whereas all Astra, Luna, and Terra packets are observed. It is almost balanced between revision stages (13 missing at revision 1 and 14 at revision 2). Consequently, the reliable agreement coefficient for the observed subset does not justify a planning–outcome conclusion.

Reactive-episode context. Episodes with further result-driven action start from higher-scoring parents on average (32.94 versus 21.33 held-out RHAE) and use more updater sessions (1.67 versus 1.11). Within the ten trajectories whose two revisions receive different adaptation labels, the reactive revision has lower change in eight. These descriptive differences do not isolate the effect of repair from the problems that prompted it.

G Campaign Integrity and Recovery Audit

The experimental protocol and coding bootstrap remained fixed throughout the campaign. A post-collection source audit confirms unchanged prompt, feedback, qualification, game-interface, scoring, and resource-control implementations across the recorded versions. Operational recovery and provenance maintenance account for changes in execution-support code; the artifact supplies a concise source change log. Retained-record checks cover all 27 complete submitted lineages. Table 18 summarizes these checks; the supplementary artifact provides pseudonymous audit exports and an offline consistency checker. These checks establish preservation and consistency of recorded controls, not an independent experimental rerun.

Table 18: Post-collection integrity checks against retained original records.

Control	Verified result
Frozen source	All 81 original snapshot fingerprints and file counts match, covering 1,985 files.
Evaluated identity	All 2,025 selected evaluations match the corresponding snapshot, runtime model-profile fingerprint, environment fingerprint, and configured output cap.
Prompts and bounds	Each rendered prompt is constant across replicates within each of nine conditions; all 27 lineages share one resource-limit profile.
Hosted validation	All 81 validations are complete, cover 25 games, agree with recorded scores, match snapshot identity, and use the same trace identity as local validation.
Coding preservation	All 54 blinded packets match their manifests; their labels and rationales are unchanged from the hash-verified frozen coding passes and consensus.

Execution recovery. Construction replacement and operational recovery are distinct. Recovery finishes an evaluation of the same frozen harness, using compatible journals for resumption when available. The controller also permits a fresh execution without a replay parent; such a restart can draw new model responses and is not deterministic resumption.

The selected-suite manifests record 2,130 attempts for 2,025 game–snapshot slots across 162 suites. Sixty-three slots have multiple attempts. Each slot contains exactly one clean attempt, which is selected; the records show no comparison of multiple clean completed scores to select a higher one. Of the 105 other attempts, 49 are classified as provider infrastructure failures, 11 as blocked, and 45 as incomplete without an outcome reason. Twenty-one slots contain a fresh restart; seventeen selected attempts are those fresh restarts. No resource-stop record was found among these 2,130 attempts. These classifications do not fully establish the cause of every interruption. The released audit contains the attempt histories and detailed categories. It covers manifest-listed histories and does not estimate a causal effect of recovery on scores.

Recovery costs. Endpoint solving cost includes fresh and replayed inference in the selected attempt. The program lower bound sums publication-priced fresh selected-game inference, observed construction/revision/repair inference, and smoke qualification. It excludes unselected game-attempt recovery overhead; unresolved terminal requests are not imputed. Operational recovery meters have not been fully reconciled to the publication pricing schedule. The reported lower bound should therefore not be interpreted as an exhaustive bill for the campaign.

Bounded frame exception. The validator accepts a narrowly specified frame-hash discrepancy caused by unseeded visual noise in `lf52`, while requiring agreement on state, level completion, and available actions. The exact predicate and expected and observed frame hashes are retained in the audit. Hosted validations contain 213 such frame events, all on `lf52`; local validation records the same event count. All 81 hosted validations report score agreement. This exception concerns visual observations, not a relaxed score criterion.

Auditable process examples. The supplement includes the frozen per-code rationales for both coding passes and consensus across all 54 episodes. Three excerpts were selected by fixed rules using packet IDs and labels, without downstream outcomes, to illustrate agreement, disagreement, and missing evidence. Each includes ordered evidence anchors, labels, rationales, and the consensus rule. Selection rules and compression markers are preserved in the artifact. These examples support inspection of the coding method; they are not complete transcripts or human validation.

Artifact availability. The supplementary review artifact contains the cleaned framework, frozen prompts and splits, condition and pricing metadata, all 81 anonymized source snapshots with original and review-copy fingerprints, trajectory and game tables, coder and consensus labels with their rationales, three chronological coding examples, two revision case studies, integrity and recovery audits, the capability manual, deterministic analyses, and a checksum manifest. Offline checks verify configuration, source fingerprints, audit consistency, labels, and example selection. The release supports reproducing the reported analyses, inspecting harness changes, and testing new methods for autonomous improvement.