# THE REVIEW TAX: FROM AUTOMATIC REVIEW TO POST-EDIT CONTROL IN CODING AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

After a coding agent produces an edit that can be applied, should the harness automatically spend another model call reviewing it? We study this *review tax*: computation committed to review simply because a patch exists, before there is evidence that review is worthwhile. In a controlled three-call harness on 60 SWE-bench Verified tasks under the same 32k context limit, review consumes 20.8% and 23.7% of all model tokens for two Qwen coder configurations. More than 83% of those review tokens are spent reading the input. Measuring the return requires care, since separately rerun review and no-review arms can produce different patches before review begins. When we instead fork the same saved patch on 200 tasks at 64k, review gains 5.0 percentage points, with a 95% interval of $[1.4, 8.6]$, while a prompt variant gives a smaller estimate whose interval includes zero. These findings motivate treating the stage after an edit as a decision. In a separate sequential campaign with 308 task entries from 305 SWE-bench-Live issues, a learned policy resolves 6.5 points more than a state-blind schedule, at higher reported cost. A further repetition again favors learned control over the schedule but leaves its advantage over a correctness gate unresolved. When a public-test result has already been obtained and charged for, showing it to later decisions yields 15.5 points higher resolution than masking it. An applicable edit is therefore an opportunity to review. Whether to do so should depend on the candidate patch, the evidence already available and the cost of continuing.

## 1 INTRODUCTION

A coding agent that has already produced a candidate patch still faces a consequential choice: should it stop, or spend more computation reviewing what it just wrote? The harness we study answers this question automatically. Whenever an edit applies, it issues another model call that rereads the issue and the edited files and either proposes another change or returns `DONE`. The trigger observes only that an edit exists. It does not ask whether the patch is likely correct, whether review is likely to repair it, whether a test would be more informative, or what another call will cost. We call the computation committed by this rule the *review tax*. An applicable edit makes review possible, but it does not by itself make review worthwhile (Figure 1).

This distinction matters because the tax is coupled to the agent's own intermediate behavior. A model that produces applicable edits more often is automatically sent to review more often, so an improvement in edit production can redirect computation into an additional stage. Each such call pays again to read the issue and edited files even when the reviewer makes no further change. The return of self-repair depends on feedback and budget (Huang et al., 2024; Olausson et al., 2024), and agentic review with repository exploration and executable checks can improve resolution (Wang et al., 2026a). Runtime controllers and learned stopping or allocation already choose additional computation from inputs or intermediate states (Damani et al., 2025; Wang et al., 2026c; Dong et al., 2026). Our question is narrower: *what does automatic post-edit review actually cost and buy, and what should determine the next action once a patch exists?*

We first measure the review tax directly, in a controlled harness that supplies the files to edit, so that once an edit applies, review is the only model computation before the tests. On 60 SWE-bench Verified tasks under a matched 32k context limit, automatic review fires on average for 45.7 tasks with Qwen2.5-Coder-32B and 55.7 with Qwen3-Coder-30B-A3B. Review consumes 20.8%



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.

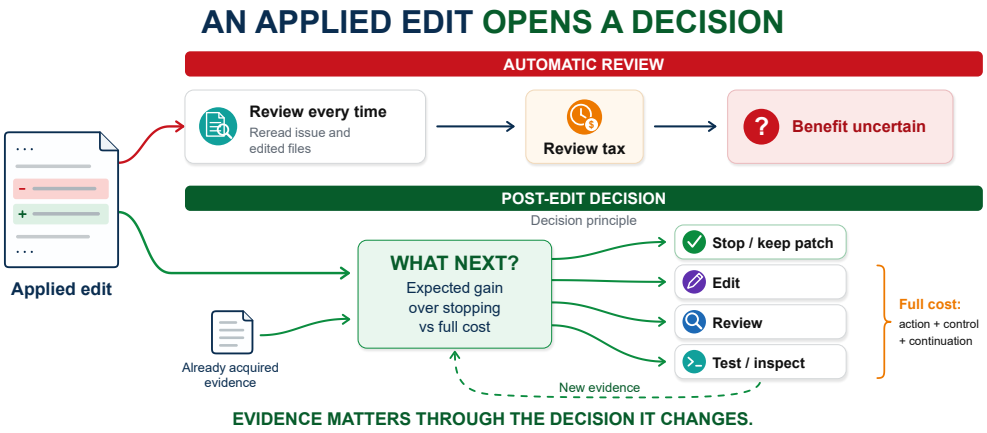


Figure 1: **From automatic review to post-edit control.** Top: an applicable edit automatically triggers review, committing tokens before its value is known. Bottom: a controller can retain the patch, edit, review or acquire evidence. Our experiments separately measure the trigger's expenditure, review's effect on a shared patch, and the outcomes and costs of sequential policies. The diagram is conceptual.

and 23.7% of all model tokens used by the harness, respectively. More than 83% of its tokens are prompt input: every review call pays to read the issue and the edited files, whatever the reviewer subsequently does. The gate thus commits a substantial fraction of computation before the value of the next review is known.

To measure review's effect on a particular candidate, we need a comparison that holds the patch fixed. Independently rerunning the harness with and without review can produce different patches before review is reached, so a difference in final outcomes need not be the effect of reviewing one candidate. In the repaired 30B-A3B comparison, 19 of the 20 discordant pairs on reviewed tasks had already diverged at that point. We therefore fork the same saved patch into review and retain branches. On 200 eligible SWE-bench Verified tasks with the 30B-A3B model at 64k, the original review prompt raises resolution from 38% to 43%, a gain of 5.0 percentage points with a 95% interval of $[1.4, 8.6]$. A variant that aligns the system and user instructions gives $+2.5$ points $[-0.4, 5.4]$. Review therefore improves resolution in this shared-patch comparison under the original prompt, while the benefit of the variant remains unresolved.

A positive average effect still leaves the harness to decide whether to review the patch in front of it. Once a patch exists, the harness can retain it, continue editing, review and revise it, or acquire evidence before deciding. We therefore compare complete post-edit policies. In a separate sequential campaign with its own worker and 308 task entries from 305 SWE-bench-Live issues, a learned policy that uses the current patch and the evidence so far resolves 46.4% of tasks, compared with 39.9% for a state-blind schedule that sees only the step, remaining budget and action feasibility. The difference is 6.5 percentage points with a paired interval of $[3.6, 9.7]$, but the learned policy also incurs higher reported charges. A further repetition on the same issues again favors it over the schedule, while leaving its advantage over a correctness gate or a policy that tests whenever eligible unresolved.

Evidence itself matters through what happens next. When the same public-test result has already been obtained and charged for, showing it to later decisions yields 15.5 percentage points higher resolution than masking it on 84 saved states, with an interval of $[7.1, 23.8]$. Since both arms have already paid for the test, this comparison measures the value of using its output. A separate comparison includes the cost of acquiring it. Together, they distinguish whether evidence can improve a later decision from whether buying that evidence justifies its cost.

We make three contributions. First, we quantify the review tax of an applied-edit trigger and show how edit applicability commits the harness to input-heavy computation (Section 3). Second, we separate that expenditure from review's effect on a particular patch, using shared-patch forks to measure

the local benefit that ordinary reruns can obscure (Section 4). Third, we compare sequential choices after an edit and show that an already acquired test output can improve downstream outcomes when later decisions can use it (Section 5).

## 2 SETUP: AUTOMATIC REVIEW AFTER AN APPLIED EDIT

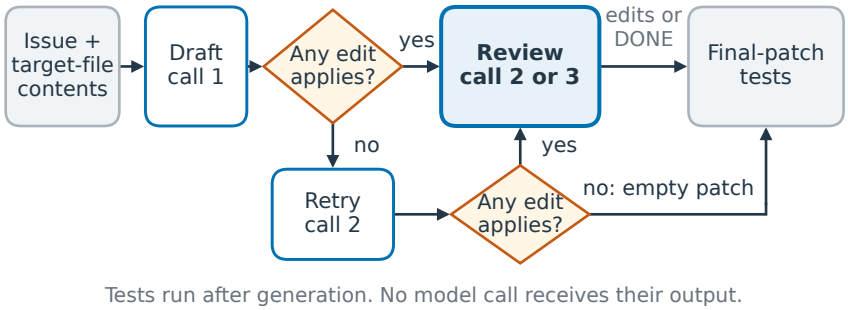


Figure 2: **The three-call harness and its applied-edit gate.** Arrows show execution order. Retry runs only if no draft edit applies. An edit that applies after either call triggers review, as call 2 or 3. Review returns further edits or `DONE`. The no-review arm omits the highlighted stage and tests the current patch. Tests provide no feedback to model calls.

**Harness.** To measure what automatic review costs and buys, we use a harness that makes at most three model calls per task (Figure 2). The draft call receives the issue and the pre-edit contents of the files touched by the benchmark's reference patch, but not the replacement text. It answers with SEARCH/REPLACE edit blocks. If no block applies, a retry call receives the failed response and asks for exact search text. Once any edit applies, a review call rereads the issue and edited files and returns further edits or `DONE`. Repository tests run only after generation, so no call sees their output. Every call requests greedy decoding with a fixed seed and at most 4,096 output tokens. Repeated model outputs therefore measure serving variability rather than deliberate sampling, while evaluator variability is measured separately. Appendix A gives the prompts.

**Why a fixed three-call harness.** The fixed stages make the review decision easy to locate: once an edit applies, review is the only additional model computation before testing. We can therefore measure how often review runs and what it consumes, then fork the same patch to estimate what review changes. This isolates the review question from other sources of agentic variation while retaining the familiar self-correction pattern of an initial output followed by critique or debugging and an opportunity to revise (Huang et al., 2024; Madaan et al., 2023; Olausson et al., 2024; Chen et al., 2024). Fixed pipelines can also be competitive with autonomous agents. Agentless uses fixed localization, repair and validation stages and reports strong performance at low cost (Xia et al., 2025), while cost-controlled evaluations find that simple baselines can outperform more complex agent architectures in some settings (Kapoor et al., 2025).

Supplying the target files holds localization fixed, so differences in repository search do not enter the review comparison. This follows SWE-bench's oracle-retrieval setting and the use of perfect localization in program-repair evaluation (Jimenez et al., 2024; Xia et al., 2023; Sepidband et al., 2026). The oracle operates at the file level: the model receives the pre-edit contents of the files touched by the reference patch, subject to the truncation limits in Appendix A, without being told which functions, lines or edit locations to change. SEARCH/REPLACE blocks follow Aider's diff edit format (Aider, 2026), making edit applicability an explicit and checkable event.

**The applied-edit gate.** The harness invokes review whenever an edit applies, after either the draft or a retry. We call this rule the *applied-edit gate*. It observes that a patch exists, but does not assess whether the patch is correct, whether review would repair it, or whether testing or stopping would serve the task better. The *review tax* is the computation committed by this rule, whether or not review proves useful. Section 3 measures the tax, Section 4 asks what it buys, and Section 5 compares policies that choose the next action after an edit.

**What we measure.** We compare the full harness, or *review arm*, with an otherwise identical *no-review arm* that omits review. This component ablation is commonly used to evaluate agent scaffolds (Yang et al., 2024). We write $K$ for the full harness and $-b$ for the harness without a behavior $b$, which in the main text is review. Let $Y_w^K, Y_w^{-b} \in \{0,1\}$ be the outcomes of task $w$ in the two arms, and let $T_b$ contain the $n_b$ of $N$ tasks on which $b$ fired in the review arm. The *exposure* of $b$ is $e_b = n_b/N$. For $n_b > 0$, its per-firing outcome contrast is

$$d_b = \frac{1}{n_b} \sum_{w \in T_b} (Y_w^K - Y_w^{-b}) = \frac{S_b - G_b}{n_b}, \tag{1}$$

where the salvages $S_b$ count tasks in $T_b$ solved only with review and the regressions $G_b$ count those solved only without it. The arms run separately and can produce different drafts before review. Thus $d_b$ compares two policies on the tasks exposed to review. We separately estimate the effect of reviewing a particular patch by holding that patch fixed (Section 4).

To measure cost, let $C_b$ denote the input plus output tokens spent on stage $b$ in the review arm, and $C_K$ the arm's total tokens. The review tax is $C_R$, the review-stage $C_b$, which we report per task and as a share $C_R/C_K$ of the total. Writing $\bar{c}_b$ for mean stage tokens per firing gives $C_b/N = e_b\bar{c}_b$: expenditure depends on both how often the stage runs and how much each call consumes. Stage tokens measure the computation committed to review. The change in total tokens after removing review is a different quantity, as are latency and money.

**Studies, tasks and models.** The primary slice contains 60 SWE-bench Verified tasks (Jimenez et al., 2024) from the 194 instances of the under-15-minute tier, fixed before any measurement. We compare Qwen2.5-Coder-32B and Qwen3-Coder-30B-A3B at a matched 32k context limit, with Llama-3.1-8B and Llama-3.3-70B providing a second model family. A harder slice contains 30 tasks from the 261 instances of the 15 to 60 minute tier, with two replicates per Qwen model. These comparisons concern model configurations because architecture, training and serving all change with model identity (Appendix B). A task is resolved when its repository-test evaluation passes. Review exposure and token counts can be measured independently of that outcome (Appendix E.4).

The shared-patch and sequential-control studies use their own cohorts and budgets to ask what should happen once a patch exists. Shared-patch review uses 200 eligible tasks at 64k. Sequential control uses a Qwen3.6-35B-A3B worker, up to six decisions and a primary $0.45 cap. Because these protocols differ from the three-call harness, their results do not estimate the savings from removing its gate. Table 5 in the appendix gives the populations and interventions.

## 3 THE REVIEW TAX

**The trigger determines exposure.** The applied-edit gate links review expenditure to the model's ability to produce an applicable edit. Every task with an edit after drafting and any retry is sent to review, regardless of whether the final patch passes tests. The tax therefore depends on how often the gate fires and how much each call consumes, as expressed by $C_R/N = e_R\bar{c}_R$ (Proposition 1). Both quantities can be measured before interpreting review's effect on resolution.

At the same 32k context limit, Qwen2.5-Coder-32B invokes review on 45/46/46 tasks across three replicates, while Qwen3-Coder-30B-A3B invokes it on 55/56/56 (Figure 3). On average, review runs on 45.7 and 55.7 of 60 tasks, giving exposures of 76.1 % and 92.8 %. The configuration that produces applicable edits more often also commits computation to review more often. Architecture, training and serving all change with model identity, so this comparison does not isolate which model property accounts for the difference.

The same pattern appears in the Llama family: Llama-3.3-70B invokes review on 34/33/34 tasks, compared with 18/19/18 for Llama-3.1-8B at 32k. On the harder 30-task tier, the two Qwen configurations invoke review on 19/20 and 24/25 tasks. Repeating these fixed slices measures exposure across runs. It does not produce independent samples of 180 or 60 tasks (Appendix D.1).

**The committed tokens are substantial and mostly input.** The matched Qwen runs spend 3 220 and 3 800 review tokens per task-replicate, or 20.8 % and 23.7 % of all model tokens in the review arm. More frequent review accounts for the larger expenditure, even though mean tokens per review call fall from 4 231 to 4 096. Each call pays again to read the issue and edited files, and input

accounts for 83.3 % and 83.8 % of the stage totals. Counting only generated tokens would therefore omit most of the review tax.

The reviewer returns `DONE` on 46.7 % and 41.3 % of calls, respectively. On many tasks it proposes further changes. The question is whether those changes improve resolution enough to justify the computation spent on review.

Separate executions without review can also differ in their drafts and non-review costs, so stage expenditure alone does not estimate the savings from removal (Appendix D.2). Monetary cost further depends on caching and prices. Applying the recorded cached prices reduces the two review-stage charges from $2.01 and $2.20 to $1.08 and $1.20 over their 180 task-replicates. These are calculated charges rather than observed invoice savings. Input also dominates the HAL trace summary, with a median input share of 95.0 % across 373 runs, although that summary covers complete agent runs rather than review alone (Appendix B.5, Kapoor et al., 2026).

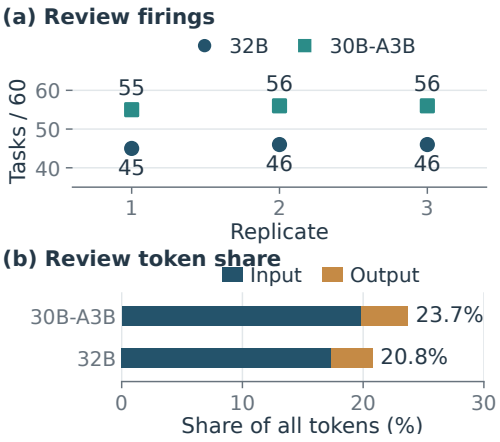


Figure 3: **Review expenditure at 32k.** Both models repeat the same 60 tasks three times. (a) Review firings. (b) Review input/output shares of all model tokens in the review arm.

## 4 What automatic review buys

**Separate reruns estimate a policy contrast.** Does the computation spent on review improve the final patch? We first compare separate executions of the harness with and without review. For Qwen3-Coder-30B-A3B at 64k, the corrected historical comparison gives $-5.0$ percentage points over 160 review firings. A new comparison with three replicates gives $+7.4$ over 162 firings. Pooling all six replicates gives $+1.2$ with a task-cluster interval of $[-3.0, 5.3]$ (Table 1A). The new runs favor review, but the pooled result remains compatible with both benefit and harm. Appendix E gives the historical comparisons for the other models and the precision extensions.

Table 1: **A rerun contrast and a same-draft effect answer different questions.** All rows use Qwen3-Coder-30B-A3B at 64k. Contrasts and supplied task-cluster intervals are in percentage points. Block A counts review-fired task-replicate pairs on E60, including reuse of the same tasks. Its pooled row is a combined analysis, not another replication. Block B uses one eligible saved state per task on a distinct 200-task cohort, with all assigned evaluations completed. Salvages and regressions count gains and losses relative to each row's reference.

| Comparison | Pairs | Salvages | Regressions | Contrast (pp) | 95% interval (pp) |
|---|---|---|---|---|---|
| **A. Separate reruns: review minus no review** | | | | | |
| Repaired historical trio | 160 | 6 | 14 | $-5.0$ | $[-10.2, +0.7]$ |
| New replication trio | 162 | 17 | 5 | $+7.4$ | $[+1.8, +13.2]$ |
| Pooled six replicates | 322 | 23 | 19 | $+1.2$ | $[-3.0, +5.3]$ |
| **B. Shared draft: review minus retain** | | | | | |
| Original prompt | 200 | 12 | 2 | $+5.0$ | $[+1.4, +8.6]$ |
| Harmonized prompt | 200 | 7 | 2 | $+2.5$ | $[-0.4, +5.4]$ |

**Holding the draft fixed isolates local review value.** An outcome difference need not be the effect of reviewing the same candidate. Separate executions can produce different drafts before review, even under greedy decoding. Let $D_w^K$ and $D_w^S$ be the drafts in the review and no-review arms, and $Y_w(D, r)$ the outcome of draft $D$ with review when $r = 1$ and without it when $r = 0$. Adding and

subtracting $Y_w(D_w^K, 0)$ separates the two sources of outcome differences:

$$d_b = \frac{1}{n_b} \sum_{w \in T_b} (\tau_w + \delta_w), \qquad \tau_w = Y_w(D_w^K, 1) - Y_w(D_w^K, 0), \qquad \delta_w = Y_w(D_w^K, 0) - Y_w(D_w^S, 0). \tag{2}$$

Here $\tau_w$ is the effect of reviewing the same draft, while $\delta_w$ is the contribution of the drafts differing before review. This distinction matters in the repaired historical comparison: 19 of the 20 pairs with different outcomes on the review-firing set already have different draft bytes. The remaining pair is a same-draft regression. The rerun contrast therefore cannot attribute its paired changes to reviewing one fixed patch (Appendix E.6). Even identically distributed drafts do not resolve this issue when selection into $T_b$ depends on the review arm's draft (Proposition 2).

**Review improves resolution under the original prompt.** To isolate review's local effect, we fork the same saved intermediate patch into review and retain branches. The study uses one eligible state for each of 200 previously unused SWE-bench Verified tasks, with Qwen3-Coder-30B-A3B at 64k. It compares the original review prompt with a harmonized prompt whose system and user instructions both allow an unchanged response. A sham branch issues the review request but discards its edits. All 200 fork roots are byte-identical across branches, every outcome is available, and every sham outcome agrees with retention.

Original review repairs 12 failed candidates and damages two successful candidates, raising resolution from 38 to 43 %, or $+5.0$ points $[1.4, 8.6]$. The harmonized prompt gives $+2.5$ points $[-0.4, 5.4]$. Review therefore has positive local value under the original prompt. The variant's unresolved effect does not establish a difference between prompts. The cohort and 64k setting differ from the matched-context cost study, so these results cannot be combined into one cost-effectiveness estimate.

**What should determine whether to review?** A positive average effect leaves open which candidates warrant review and whether the computation could be better spent elsewhere. On 200 tasks, a quality-based trigger gains $+3.5$ points $[0.3, 6.7]$ over random invocation, but its adjusted exact $p = 0.196$ leaves the comparison unresolved. Four alternative allocations under the same 16 000-token cap each yield $0.0$ points $[-2.0, 2.0]$ relative to original review. Neither comparison establishes a better allocation rule. The local benefit also remains unresolved on other cohorts: review gains $+1.3$ points $[-7.4, 10.0]$ on 150 Live tasks and $+1.7$ points $[-13.4, 16.8]$ on 60 ProMax tasks.

Two further studies clarify where variation and useful evidence enter. Repeating full generation yields different outcomes in 51 of 120 pairs, compared with zero of 240 frozen-prefix continuations, reinforcing the value of comparing continuations from the same candidate. Supplying public-test execution and its feedback improves resolution by $+9.0$ points $[4.1, 13.9]$ over repository exploration alone on 200 tasks. Thus the evidence available during review can change its value. Appendix F gives the controls and full contrasts for these six studies, E1 through E6.

**Review at an agent's finish decision.** Does this picture extend beyond the three-call harness? In OpenHands, a functioning critic evaluates the agent's finish claim. Across two critic configurations and two repetitions on the same 59 tasks, review-enabled policies resolve three or four more tasks per cell, gains of 5.1 to 6.8 percentage points. The critic consumes 540 to 564 tokens per firing. These separate policy executions show higher observed resolution in this setting without isolating a same-patch review effect. The four cells reuse the same tasks and do not establish a general benefit of review (Appendix J).

Review can therefore repair a fixed candidate, and executable feedback can make it more useful. A rerun comparison alone can obscure that value by mixing review's effect with earlier variation. The remaining question is how to choose among review, another edit, acquiring evidence and stopping, while accounting for the cost of each choice.

## 5 Post-edit control: choosing the next action

If an applicable edit does not by itself justify review, what should the harness do instead? It can retain the patch, continue editing, review and revise it, or acquire evidence before deciding. Each choice changes what the harness can do next, so the relevant comparison is between complete post-

edit policies. We study these policies in a separate sequential campaign on SWE-bench-Live (Zhang et al., 2025).

**Valuing the next action.** For state $x$, outcome $Y \in \{0, 1\}$ and charge $C$, define the incremental value and cost of action $a$ followed by a fixed continuation $\pi_0$ relative to stopping:

$$\tau_a(x) = \mathbb{E}[Y^{a,\pi_0} - Y^{\text{stop}} \mid x], \qquad \kappa_a(x) = \mathbb{E}[C^{a,\pi_0} - C^{\text{stop}} \mid x]. \tag{3}$$

Correctness alone does not determine this value: an incorrect patch may resist repair, and further editing can damage a correct patch (Proposition 3). We therefore evaluate a controller that scores actions by predicted gain and cost. Its six actions are stop, edit, review with revision, execution review with revision, repository inspection and public testing. The controller uses scores $\widehat{\tau}_a - \lambda\widehat{\kappa}_a$ and samples from a softmax over feasible actions passing a predicted-damage filter. It uses $\lambda = 4$, temperature $0.25$ and risk threshold $0.05$, with at most six decisions and a \$0.45 episode cap. The filter does not certify a harm bound. Replaying the supplied weights, features and random draws reproduces all 8 323 decisions with recorded score inputs across the source, confirmation, API and myopic arms, plus all 168 disclosure first decisions (Appendix G).

Table 2: **Resolution and cost of the primary policies and follow-up continuations.** The first two rows start from the same checkpoints. The remaining rows begin later in the trajectories, so their differences combine the choice of policy with the change in starting state. Charges follow the reported follow-up account, whose boundary differs from the original episode account (Appendix G). All rows retain 308 assigned entries from 305 canonical issues.

| Policy | Solves / 308 | Resolution (%) | Mean charge (\$) |
|---|---|---|---|
| **A. Primary policies from the same checkpoints** | | | |
| Learned, $\lambda = 4$ | 143 | 46.4 | 0.123 |
| State-blind schedule | 123 | 39.9 | 0.107 |
| **B. Follow-up continuations from later checkpoints** | | | |
| Evidence-masked continuation | 126 | 40.9 | 0.121 |
| Always-review continuation | 130 | 42.2 | 0.141 |
| Learned, selected cost weight | 137 | 44.5 | 0.105 |

**Selective control improves on a state-blind schedule.** With Qwen3.6-35B-A3B as the coding worker, we assign 308 task entries representing 305 canonical issues to each policy. The state-blind schedule sees only the decision step, remaining budget and action feasibility. The learned policy resolves 143 entries against its 123, a gain of $6.5$ percentage points $[3.6, 9.7]$, at reported mean full-policy charges of \$0.123 and \$0.107 (Table 2). The gain therefore comes with higher expenditure. Both the follow-up account and the original episode account agree on that ordering, although their totals differ (Appendix G). Does the advantage recur beyond this execution? A repeat on the source tasks gains $+3.9$ points over the schedule, and an additional cohort of 250 issues gains $+6.4$ points. Table 3 reports their uncertainty and mean charges alongside the stronger reference policies. Each paired comparison shares its starting state, and the gains over the schedule again come at higher cost.

**The role of evidence.** Masking evidence features leaves 126 solves, only three more than the schedule's 123, compared with twenty more for the primary learned policy. Always review yields 130 solves, and a selected cost weight yields 137 at a reported charge of \$0.105. These follow-up policies start from later checkpoints than the original policies, so the $5.5$-point masking difference and $4.2$-point always-review difference combine changes in both policy and starting patch. They suggest where evidence and action selection may matter, but do not isolate their effects or establish cost dominance. To test evidence use directly, we must hold the starting state fixed and change only what the controller sees.

**Using evidence and acquiring it are different questions.** For an acquired test output $o$, the action's value decomposes as

$$\tau_{\text{test}}(x) = \underbrace{\mathbb{E}[Y^{\text{test},\pi_0(o)} - Y^{\text{test},\pi_0(\varnothing)} \mid x]}_{\text{using the output}} + \underbrace{\mathbb{E}[Y^{\text{test},\pi_0(\varnothing)} - Y^{\text{stop}} \mid x]}_{\text{continuing without the output}}.$$

Table 3: **The schedule advantage recurs, while stronger references remain competitive.** All comparisons use the \$0.45 cap. The source repeat has 308 entries from 305 issues, and the additional cohort has 250 distinct issues. Intervals are pointwise 95 % paired canonical-issue bootstrap intervals. Charges use the confirmation episode ledger. Two learned endpoints per cohort are unavailable and retain their assigned denominator.

| Comparison | Solves | | Difference (pp) | Mean charge (\$) | |
|---|---|---|---|---|---|
| | Learned | Reference | [95% interval] | Learned | Reference |
| **A. Learned versus state-blind schedule** | | | | | |
| Source repeat ($n = 308$) | 123 | 111 | +3.90 [0.97, 7.12] | 0.1247 | 0.1094 |
| Additional cohort ($n = 250$) | 106 | 90 | +6.40 [2.40, 10.40] | 0.1211 | 0.1073 |
| **B. Source repeat versus stronger references** ($n = 308$) | | | | | |
| Correctness gate | 123 | 125 | −0.65 [-4.21, 2.91] | 0.1247 | 0.0732 |
| Test-all | 123 | 122 | +0.32 [-3.26, 3.90] | 0.1247 | 0.1304 |

The reveal/mask experiment isolates the first term. Both branches start after the same public test, pay its \$0.09 acquisition charge and cannot acquire it again. Only the reveal branch receives the output, as verified by the controller inputs and isolation checks. Revealing changes the first action in 34 pairs and resolves 47 of 84 states against 34 for masking, a 15.5-point gain $[7.1, 23.8]$. Thus an observation already paid for can still have substantial value through the decisions that follow. A separate acquisition comparison resolves 47 of 82 states with the test against 33 without it, +17.1 points, at an additional mean charge of \$0.091. Using evidence and paying to acquire it are therefore distinct decisions (Figure 4).

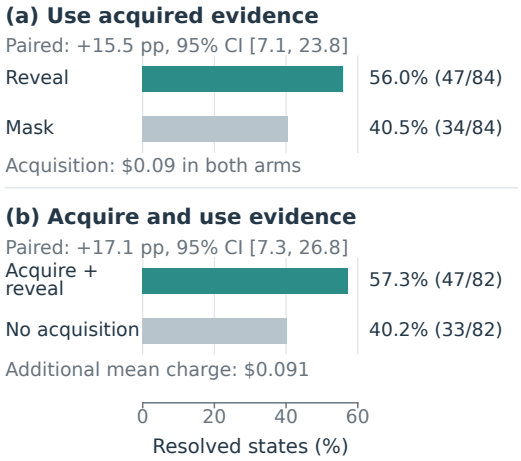


Figure 4: **Using and acquiring evidence.** The panels show separate paired comparisons with pointwise 95 % bootstrap intervals. (a) Acquisition is fixed. (b) The comparison includes acquisition and its cost.

**The advantage depends on the reference policy.** The source repeat does not resolve an advantage over the correctness gate or test-all (Table 3). The correctness gate resolves 125 entries against the learned policy's 123 at mean charges of \$0.0732 and \$0.1247, respectively. Thus a correctness-based rule remains a competitive reference at lower recorded expenditure. Under three pinned API workers, the learned policy exceeds its schedule by 10 to 11 points at higher mean charges, while differences from their correctness gates remain unresolved. The evidence supports learned control over a state-blind schedule and shows that disclosed observations can improve subsequent decisions. It leaves open whether learned incremental-value control is preferable to correctness-based control. The prediction diagnostics assess terminal resolution rather than incremental gain, so the policy gains do not establish calibration of the action-value estimates.

## 6 Related work

**Self-review, self-repair and feedback.** Review-then-revise loops feed a critique back into generation (Madaan et al., 2023; Shinn et al., 2023). Their return depends on the feedback and comparison budget: intrinsic self-correction can degrade answers (Huang et al., 2024), and code self-repair gains shrink against equal-cost resampling (Olausson et al., 2024). Execution is not uniformly helpful either. An untrained agent's tests lower resolution while a stronger model's tests raise it (Tao et al., 2026), and prohibiting execution in commercial repair agents changes resolution by a nonsignificant 1.25 points on average while saving over half of one agent's tokens (Lin et al., 2026). Conversely,

review with repository exploration and executable checks outperforms single-turn, fixed-context, feedback-free review like ours (Wang et al., 2026a). These findings motivate measuring what a particular review stage costs and buys.

**Harness design and replay-based attribution.** Harness design changes outcomes and token cost even for a fixed model (Sghaier et al., 2026; Vats & Golev, 2026), with input tokens driving much of the cost of agentic coding (Bai et al., 2026). We trace review-stage tokens to the gate that commits them, then ask what return that expenditure produces. A shared candidate lets us separate review's effect from changes earlier in generation. Replay studies reconstruct a shared prefix before intervening (Shah, 2026; Gonuguntla, 2026), and replayed multi-agent debugging shows no advantage for feedback-based repair over plain rerunning at equal budget (Luan et al., 2026). Our shared-patch fork applies this principle to automatic post-edit review, whose separately rerun arms can otherwise reach different patches before the intervention (Section 4).

**Adaptive allocation, verification and stopping.** Input-adaptive allocation (Damani et al., 2025; Zhai et al., 2026) and cost-sensitive gating of tests, critics and verifiers (Ding et al., 2026; Papamarkou et al., 2026) already treat added computation as a decision. Learned controllers choose when to check (Yi & Song, 2026), steer a fixed coding worker (Wang et al., 2026c) and transfer from smaller generators to larger commercial models (Wang et al., 2026b). Learned stopping can abort local runs (Pham et al., 2026) and beat scalar rules in some regimes (Dong et al., 2026), while prior-run experience can end patch generation early (Guo et al., 2026). Our question concerns the computation triggered once an edit applies. We connect its cost to the local effect of reviewing the same patch, then compare complete post-edit policies against a schedule that sees only the step, remaining budget and action feasibility. The reveal/mask comparison further separates using an already acquired test output from acquiring it (Section 5). Appendix I gives the extended comparison.

## 7 Discussion and conclusion

**Review has value, but applicability alone does not determine it.** The applied-edit gate commits computation whenever an edit applies, whether or not the patch would benefit from review. At matched 32k context, this stage consumes 20.8 % and 23.7 % of all tokens in the two Qwen configurations, mostly prompt input. We measure review's local effect by holding the candidate fixed. Independent reruns can diverge before review, while the 200-task shared-patch study finds a 5.0-point gain for the original prompt. The harmonized prompt leaves its gain unresolved, and execution feedback provides another setting in which review helps. Review's value should therefore be stated for the prompt, feedback and candidate distribution studied. The practical question is when and how to review a patch that already exists.

**Post-edit control must value both actions and information.** The learned policy resolves more tasks than a state-blind schedule in the primary campaign, the source repeat and the additional cohort, at higher charges. The reveal/mask intervention shows why evidence matters: showing an already acquired test result changes subsequent decisions and improves resolution. Acquiring that result remains a separate decision with its own cost. The evidence-masked, cost-weight and always-review follow-ups suggest further ways to allocate computation, but their later starting checkpoints and differing source cost accounts prevent a controlled claim of better cost efficiency. The source repeat also leaves comparisons with the correctness gate and test-all unresolved. The supported advantage is over a state-blind schedule, with a separate demonstration that later decisions can benefit from disclosed evidence.

**Scope and remaining questions.** The primary harness supplies target files, makes at most three calls and withholds test feedback until final evaluation. The sequential and functioning-critic studies broaden the setting, but the primary harness's token shares need not carry over to agents that locate their own files. Repeated tasks, pointwise uncertainty and imperfect repository tests also limit generalization. Controller replay verifies action selection from the recorded features. It does not reconstruct the frozen representation encoder or turn terminal-outcome diagnostics into calibration of incremental gain. A common-checkpoint comparison with reconciled deployment costs would test cost efficiency, and a harness that localizes its own files would test how far the findings transfer. An applicable edit creates an opportunity for further computation. What should happen next depends on what an action can improve, what evidence it uses and what its continuation costs.

### Reproducibility statement

We will publish the code and supporting artifacts upon acceptance. Appendix A gives the primary harness prompts, Appendix B specifies task populations, serving configurations and resource accounts, and Appendix C states the formal assumptions and proofs. Appendices E, F and G describe the uncertainty calculations, experimental controls and policy replay. The release will include the task manifests and per-call and per-task records needed to reproduce the reported measurements.

### Ethics statement

The study uses existing repository-task benchmarks, open-weight models served locally, and models accessed through hosted APIs. Benchmark contamination and imperfect tests limit interpretation, so passing the evaluator is not treated as a guarantee of semantic correctness or safe deployment. Reuse of models, task repositories, and artifacts remains subject to their respective licenses.

### AI use statement

Generative AI tools were used for manuscript revision and language editing. The human authors developed the hypotheses and methodology, implemented the research code, conducted the experiments, and performed the research analyses. The authors are responsible for the final content.

## References

Chris Adams, Arjun Singh Banga, Parveen Bansal, Souvik Bhattacharya, Payal Bhuptani, Rujin Cao, Pedro Canahuati, Nate Cook, Brian Ellis, Prabhakar Goyal, Gurinder Grewal, Tianyu He, Matt Labunka, Alex Manners, David Molnar, Ging Cee Ng, Vishal Parekh, Jiefu Pei, Frederic Sagnes, James Saindon, Will Shackleton, Sid Sidhu, Gursharan Singh, Karthik Chengayan Sridhar, Matt Steiner, Pratibha Udmalpet, Sean Xia, Stacey Yan, Audris Mockus, Peter Rigby, and Nachiappan Nagappan. Automating low-risk code review at meta: Radar, risk calibration, and review efficiency, 2026. URL https://arxiv.org/abs/2605.30208.

Aider. Aider documentation: Edit formats. https://aider.chat/docs/more/edit-formats.html, 2026. Documentation accessed September 7, 2026.

Anonymous. Steer, don't solve: Training small critic models for large code agents. In *Submitted to ACL Rolling Review - May 2026*, 2026. URL https://openreview.net/forum?id=ua2OJLBPbu. under review.

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Yang Wang. SWE-search: Enhancing software agents with monte carlo tree search and iterative refinement. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G7sIFXugTX.

Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. How do coding agents spend your money? analyzing and predicting token consumptions in agentic coding tasks, 2026. URL https://openreview.net/forum?id=1bUeVB3fov.

Rohith Reddy Bellibatlu, Manpreet Singh, Deepak Parashar, and Rahul Joshi. Counterfactual fairness audits of multi-step clinical llm agents require a measured per-action instability floor, 2026. URL https://arxiv.org/abs/2609.03221.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

Yihang Chen, Pin Qian, Su Wang, Chong Peng, Huan Xu, Xiyang Wu, and Yiqi Sun. Noise floor audit for agent benchmarks, 2026a. URL https://arxiv.org/abs/2608.22331.

Zhi Chen, Zhensu Sun, Yuling Shi, Chao Peng, Xiaodong Gu, David Lo, and Lingxiao Jiang. Rethinking the value of agent-generated tests for llm-based software engineering agents, 2026b. URL https://arxiv.org/abs/2602.07900.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6qUUgw9bAZ.

Wenxuan Ding, Nicholas Tomlin, and Greg Durrett. Calibrate-then-act: Cost-aware exploration in llm agents, 2026. URL https://arxiv.org/abs/2602.16699.

Sajib Acharjee Dip, Dawei Zhou, and Liqing Zhang. Think again or think longer? selective verification for budget-aware reasoning, 2026. URL https://arxiv.org/abs/2606.19808.

Zhe Dong, Fang Qin, and Manish Shah. When does learning to stop help? a cost-aware study of early exits in reasoning models, 2026. URL https://arxiv.org/abs/2606.30852.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 1097–1104, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Zou. Cost-of-pass: An economic framework for evaluating language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vC9S20zsgN.

Yong eun Cho. It's not the capability: Harness sensitivity is non-monotone across llm agent tiers, 2026. URL https://arxiv.org/abs/2605.26731.

Run-Ze Fan, Zihao Zhang, Simin Ma, Yebowen Hu, Shouju Wang, Kaiqiang Song, Fei Liu, Hamed Zamani, and Xiaoyang Wang. An empirical study of harness design for coding agents, 2026. URL https://arxiv.org/abs/2609.20804.

Constantine E. Frangakis and Donald B. Rubin. Principal stratification in causal inference. *Biometrics*, 58(1):21–29, 2002. doi: 10.1111/j.0006-341X.2002.00021.x.

Thibaud Gloaguen, Niels Mündler, Mark Müller, Veselin Raychev, and Martin Vechev. Evaluating agents.md: Are repository-level context files helpful for coding agents?, 2026. URL https://arxiv.org/abs/2602.11988.

Ashritha Gonuguntla. The replay gap: Static evaluation of model switching in LLM agents scores the wrong world. In *Efficient Reasoning Workshop at COLM*, 2026. URL https://arxiv.org/abs/2608.08239.

Yaoqi Guo, Ying Xiao, Jie M. Zhang, Mark Harman, Yiling Lou, Yang Liu, and Zhenpeng Chen. EET: Experience-driven early termination for cost-efficient software engineering agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 33008–33024, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1652. URL https://aclanthology.org/2026.findings-acl.1652/.

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

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

Haolin Jin and Huaming Chen. Are llms reliable code reviewers? systematic overcorrection in requirement conformance judgement. *Automated Software Engg.*, 33(3), June 2026. ISSN 0928-8910. doi: 10.1007/s10515-026-00638-5. URL https://doi.org/10.1007/s10515-026-00638-5.

Alibek T Kaliyev and Artem Maryanskyy. How much coordination gain is real? a paired noise-floor protocol for multi-agent LLM benchmarks. In *KDD workshop on Evaluation and Trustworthiness of Agentic AI*, 2026. URL https://openreview.net/forum?id=g9epz2aV4O.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Zy4uFzMviZ.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vUaY1t64ZZ.

Prakhar Khatri. Do context files help coding agents? a two-agent ablation study on real repositories, 2026. URL https://arxiv.org/abs/2607.27250.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Sydney Lewis. Same model, different harness: Different coding-agent results, 2026. URL https://arxiv.org/abs/2608.26218.

Zhengfeng Li, Lei Zhang, Xianwei Wu, Zhengqi Zhuang, Yingjie Xu, Boge Wang, Shaofei Zhu, Chuan Wang, Peng Zhao, Xinyu Zheng, and Guoping Rong. Opencodereview: Determinism over non-determinism for cost-effective agent-based code review, 2026a. URL https://arxiv.org/abs/2608.09290.

Ziming Li, Jiatan Huang, Xiaoguang Guo, Guilin Wang, and Chuxu Zhang. Same signal, opposite meaning: Direction-informed adaptive learning for llm agents, 2026b. URL https://arxiv.org/abs/2605.06908.

Zhihao Lin, Junhua Zhu, Mingyi Zhou, Xin Wang, Zhensu Sun, Renyu Yang, David Lo, and Li Li. To run or not to run: Analyzing the cost-effectiveness of code execution in llm-based program repair, 2026. URL https://arxiv.org/abs/2606.26978.

Ming Liu. More is not always better: Cross-component interference in llm agent scaffolding, 2026. URL https://arxiv.org/abs/2605.05716.

Quanyu Long, Kai Jie Jiang, Jianda Chen, Xu Guo, Leilei Gan, and Wenya Wang. Self-verification dilemma: Experience-driven suppression of overused checking in llm reasoning, 2026. URL https://arxiv.org/abs/2602.03485.

Zhongwen Luan, Xiaoyu Zhang, Ming Hu, Yue Yang, Jiongchi Yu, and Xiaohong Chen. Repair or resample? rethinking failure debugging in llm multi-agent systems, 2026. URL https://arxiv.org/abs/2608.25920.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Iliya Mirzaei. Sample more, reflect less: Self-refine and reflexion lose to repeated sampling at equal token cost, from 1.5b to 7b, 2026. URL https://arxiv.org/abs/2607.28576.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=y0GJXRungR.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Theodore Papamarkou, Vladislav Smirnov, Viktor Mazanov, Artem Vazhentsev, Preslav Nakov, Timothy Baldwin, and Artem Shelmanov. Bayesian control for coding agents, 2026. URL https://arxiv.org/abs/2606.24453.

Dzung Pham, Kleomenis Katevas, Ali Shahin Shamsabadi, and Hamed Haddadi. Agentstop: Terminating local ai agents early to save energy in consumer devices. In *Proceedings of the ACM Conference on AI and Agentic Systems*, CAIS '26, pp. 1051–1069, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400724152. doi: 10.1145/3786335.3813163. URL https://doi.org/10.1145/3786335.3813163.

Vikas Reddy, Sumanth Reddy Challaram, and Abhishek Basu. Reason less, verify more: Deterministic gates recover a silent policy-violation failure mode in tool-using LLM agents. In *KDD workshop on Evaluation and Trustworthiness of Agentic AI*, 2026. URL https://openreview.net/forum?id=wjrtCb4W9C.

Benjamin Rombaut. Inside the scaffold: A source-code taxonomy of coding agent architectures, 2026. URL https://arxiv.org/abs/2604.03515.

Mohamad Salim, Jasmine Latendresse, SayedHassan Khatoonabadi, and Emad Shihab. Tokenomics: Quantifying where tokens are used in agentic software engineering, 2026. URL https://arxiv.org/abs/2601.14470.

Brian Sam-Bodden. What context does a coding agent actually need to act?, 2026. URL https://arxiv.org/abs/2607.09691.

Melika Sepidband, Hung Viet Pham, and Hadi Hemmati. On the role of fault localization context for llm-based program repair, 2026. URL https://arxiv.org/abs/2604.05481.

Oussama Ben Sghaier, Hao Li, Bram Adams, and Ahmed E. Hassan. Don't blame the large language model: How agent harness evolution shapes coding agent quality, 2026. URL https://arxiv.org/abs/2607.03691.

Jaineet Shah. Causal agent replay: Counterfactual attribution for llm-agent failures, 2026. URL https://arxiv.org/abs/2606.08275.

Yuling Shi, Jinghan Xu, Kelin Fu, Wenhao Zeng, YINGWEI MA, Shilin He, Lei Zhang, Yue Liu, Zelin Zhao, Terry Yue Zhuo, Jialun Cao, Siyu Ye, Tianyu Liu, Kai Cai, Shing-Chi Cheung, and Xiaodong Gu. SWE-bench promax: Benchmarking agents on large-scale multilingual code refactoring. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=12gyjCxLXM.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Leitian Tao, Baolin Peng, Haorui Wang, Hang Wang, Hao Cheng, Wenlin Yao, Qianhui Wu, Tao Ge, Sharon Li, and Jianfeng Gao. Execcritic: Learn to test, test to improve for coding agents, 2026. URL https://arxiv.org/abs/2609.09133.

Naman Vats and Oleg Aleksandrovich Golev. The scaffold effect in coding agents: Harness choice as a hidden variable in coding-agent evaluation. In *Deep Learning for Code: Towards Human-Centered Coding Agents*, 2026. URL https://openreview.net/forum?id=nw4d83V687.

Yuvraj Verma. Try again, don't look back: Blind resampling outperforms self-repair in small code models, 2026. URL https://arxiv.org/abs/2607.26117.

Ruoyu Wang, Jierun Chen, Shaowei Wang, Chaofan Tao, Sidi Yang, Yuxin Jiang, Kim-Hui Yap, Lifeng Shang, Xiaohui Li, and Haoli Bai. Swe-review: Closing the loop on issue resolution with agentic code review, 2026a. URL https://arxiv.org/abs/2607.06065.

Xiaoxuan Wang, Haixin Wang, Alexander Taylor, Jason Cong, Yizhou Sun, and Wei Wang. Harnessbridge: Learnable bidirectional controller for llm agent harness, 2026b. URL https://arxiv.org/abs/2606.12882.

Yi Wang, Haopeng Zhang, Chengxiang Huang, Rui Dai, Kaikui Liu, Piotr Koniusz, and Xiangxiang Chu. Looparena: Benchmarking models as runtime controllers for loop engineering, 2026c. URL https://arxiv.org/abs/2608.28281.

Sarel Weinberger and Amir Hozez. Token reduction is not cost reduction, 2026. URL https://arxiv.org/abs/2607.12161.

Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. Automated program repair in the era of large pre-trained language models. In *Proceedings of the 45th International Conference on Software Engineering*, ICSE '23, pp. 1482–1494. IEEE Press, 2023. ISBN 9781665457019. doi: 10.1109/ICSE48619.2023.00129. URL https://doi.org/10.1109/ICSE48619.2023.00129.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying llm-based software engineering agents. *Proc. ACM Softw. Eng.*, 2(FSE), June 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

Zuodong Xiang, Yike Zhang, Hailu Xu, and YueMing Zhang. Cross-model LLM code review: Should you use claude to review codex or vice versa? In *KDD 2026 Workshop on Agentic Software Engineering*, 2026. URL https://openreview.net/forum?id=Fh32FqhIfG.

Chenyang Yang, Xinran Zhao, Tongshuang Wu, and Christian Kästner. Better harnesses, smaller models: Building 90URL https://arxiv.org/abs/2607.08938.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: agent-computer interfaces enable automated software engineering. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Haiwen Yi and Xinyuan Song. Learning to control llm agent harnesses with offline reinforcement learning, 2026. URL https://arxiv.org/abs/2607.05458.

Zhiyuan Zhai, Bingcong Li, Bingnan Xiao, Ming Li, and Xin Wang. Adaptive test-time compute allocation for reasoning llms via constrained policy optimization, 2026. URL https://arxiv.org/abs/2604.14853.

Chubin Zhang, Zhenglin Wan, Xingrui Yu, Pengfei Zhou, Wangbo Zhao, Jingxuan Wu, Yaxin Zhou, and Ivor Tsang. Don't blindly trust it: How unreliable feedback breaks tool-using llm agents, 2026a. URL https://arxiv.org/abs/2606.21409.

Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, Elsie Nallipogu, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. Swe-bench goes live!, 2025. URL https://arxiv.org/abs/2505.23419.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. Scaling laws for agent harnesses via effective feedback compute, 2026b. URL https://arxiv.org/abs/2605.29682.

## GUIDE TO THE APPENDICES

Appendix A gives the primary harness prompts, Appendix B maps studies to populations and compute, and Appendix C states the accounting and decision identities. Appendix D reports repeated exposure and cost measurements, while Appendix E separates rerun variability, evaluation noise and pre-review divergence. Appendix F reports shared-patch review, gate and budget comparisons, transfer and execution feedback. Appendix G specifies the deployed policy and its replay, sequential results, disclosure intervention and follow-up diagnostics. Appendix H retains the earlier policy comparison. Appendix I expands the literature comparison, Appendix J covers source triggers and the functioning finish critic, and Appendix K states the remaining boundaries. The result audit and reproducibility record retain source identifiers and calculation details outside the narrative.

## A PRIMARY HARNESS PROTOCOL AND PROMPTS

This appendix answers what each call of the primary harness sees and what each intervention changes, and it reproduces every prompt string verbatim so that the applied-edit gate, the retry, the review and the alternative arms can be checked against the text the models received. Draft, retry and review share one edit-producing interface. The listings below are transcribed from the primary harness source, in Python syntax so that quotation marks, newlines and substitutions are unambiguous. String literals are verbatim and control flow is condensed where a shorter expression yields the same messages. Adjacent literals concatenate without a newline, and the escapes `\n`, `\u2014` and `\u2026` denote a newline, the original dash character and the original ellipsis character.

### A.1 ORACLE FILE LOCALIZATION AND WHAT THE INTERVENTIONS CHANGE

The primary harness extracts target-file paths from the benchmark's reference patch and reads their pre-edit contents from the benchmark image. It does not supply the reference patch's replacement text. The draft prompt contains the issue description capped at 6 000 characters and each available target file capped at 16 000 characters. Context removal retains the target-file paths but omits their contents. Consequently, this intervention measures the provision of file contents under oracle localization, not the value of locating files autonomously.

Edits use SEARCH/REPLACE blocks. The gate condition is that at least one SEARCH block of the draft or the retry matched the current file contents, so review fires exactly on the applicability event of Proposition 1. It does not test whether the replacement changed a byte, whether the patch resolves the issue, or what a test would report, since repository tests run only after generation. An applicable edit is therefore an event about edit production and nothing more.

### A.2 INPUTS, TRUNCATION, AND THE SYSTEM MESSAGE

The listing shows how the harness applies these caps. They are character limits, separate from the model's token context ceiling, and the marker is appended only when a file is cut.

```
problem = instance.get("problem_statement", "")[:6000]
_CAP = 16000

def _body(c):
    return (c if len(c) <= _CAP else c[:_CAP]
            + "\n# \u2026 [truncated for context budget] \u2026\n")

files_blob = "\n\n".join(
    f"### FILE: {p}\n```python\n{_body(c)}\n```"
    for p, c in originals.items())
ctx = f"Files to edit (current contents):\n\n{files_blob}"
```

Every primary model call receives the following system string. The SEARCH/REPLACE examples are part of the prompt, including their angle brackets.

```
SYS = (
```

```
    'You are an expert software engineer fixing a real GitHub '
    'issue with the SMALLEST possible edit. You will be shown the '
    'issue and the current contents of the file(s) to edit. '
    'Respond ONLY with one or more edit blocks in this exact '
    'format, each immediately preceded by a `### FILE: <path>` '
    'line naming the file to change:\n\n### FILE: path/to/file.py\n'
    '<<<<<<< SEARCH\n<a short, EXACT, contiguous snippet copied '
    'verbatim from the current file>\n=======\n<the replacement '
    'snippet>\n>>>>>>> REPLACE\n\nThe SEARCH text must match the '
    'current file character-for-character (indentation included) '
    'and be just large enough to be unique. Change ONLY the lines '
    'needed to fix the issue; never rewrite or reformat the whole '
    'file (that breaks other tests). Emit multiple blocks for '
    'multiple edits. No prose.'
)
```

### A.3 Draft and context removal

The draft receives two messages and no earlier model response.

```
draft_msgs = [
    {"role": "system", "content": SYS},
    {"role": "user", "content": f"ISSUE:\n{problem}\n\n{ctx}"}]
```

In the ordinary context-removal arm only the `ctx` block changes. The target paths remain in the prompt, so the intervention removes file contents while retaining oracle localization.

```
ctx = (
    "Files to edit (contents NOT shown \u2014 infer the exact "
    "snippet from the issue):\n"
    + "\n".join(f"- {p}" for p in originals.keys()))
```

The harness source also contains a graded-context variant that appends reference-derived SEARCH anchors. No result in this paper uses it, and the context-removal arm defined above supplies no anchor.

### A.4 Retry after an unapplied draft

The harness parses the draft and attempts its edits against the original files. If none applies, retry appends the draft as an assistant message and the following user message to `draft_msgs`. Retry therefore sees both the original draft context and its unsuccessful response. A successful retry replaces the edited state used by the subsequent gate.

```
retry_msgs = draft_msgs + [
    {"role": "assistant", "content": draft},
    {"role": "user", "content": (
        "None of your edits applied \u2014 the SEARCH text must be "
        "copied VERBATIM (exact characters and indentation) from "
        "the file shown. Re-issue the SEARCH/REPLACE block(s) with "
        "exact SEARCH text.")}]
```

### A.5 Review of the edited state

Review fires when an edit applies after draft or retry. Its user message is constructed anew, rather than by appending the conversation history, and it shows only files whose bytes differ from the originals. Because the gate tests SEARCH matching rather than byte change, an edit whose REPLACE text equals its SEARCH text is exposed to review with the issue and no file contents. The reviewer receives the issue and current file contents, not the benchmark test results.

```
REVIEW_LEAD_GATED = (
    'The file(s) below already contain a proposed fix.'
)
```

```
show = [p for p in originals if edited[p] != originals[p]]
lead = REVIEW_LEAD_GATED
body = {p: (_body(edited[p])
            if 'context_reread' in behaviors_enabled
            else edited[p][:_CAP]) for p in show}
cur_blob = "\n\n".join(
    f"### FILE: {p}\n```python\n{body[p]}\n```"
    for p in show)
review_msgs = [
    {"role": "system", "content": SYS},
    {"role": "user", "content": (
        f"ISSUE:\n{problem}\n\n{lead} Verify it "
        "resolves the issue AND does not break unrelated behavior. "
        "If a correction is needed, emit additional SEARCH/REPLACE "
        "block(s) against the CURRENT contents below; if it is "
        f"already correct, reply exactly DONE.\n\n{cur_blob}")}]
```

Review truncates with the draft's marker in the arms that supply file contents. In the context-removal arm it uses the first 16 000 characters of each edited file without the marker, so the two arms' review prompts differ in that detail as well as in the draft context. Review edits apply to the current edited state. The parser treats a response with `DONE` in its first ten characters, case-insensitively, as requiring no further edit. That check is broader than the prompt's request to reply exactly `DONE`, and edit blocks that follow such an opening are not applied.

### A.6 Redraft, ungated review, and model substitution

The redraft arm replaces the review turn on the same applied-edit trigger. It reconstructs the draft from the original files and appends the following suffix. The word "independent" is part of the historical prompt, not an assumption that the resulting samples are statistically independent.

```
REDRAFT_SUFFIX = (
    'This is a second, independent attempt at the same task. '
    'Ignore any earlier attempt and produce your best '
    'SEARCH/REPLACE block(s) now.'
)
redraft_msgs = [draft_msgs[0], {
    "role": "user",
    "content": draft_msgs[1]["content"] + "\n\n" + REDRAFT_SUFFIX}]
```

If this attempt applies an edit, the arm selects its patch, and otherwise it retains the first post-retry patch. The rule never selects between patches using benchmark outcomes. Its budget check uses the whole-arm token differences defined in the alternative-allocation subsection of Appendix D.

Ungated review preserves the ordinary review message on tasks satisfying the original gate. Where no edit applies, it uses all original files in `show` and replaces only the opening clause below. It still receives the unchanged remainder of the review instruction.

```
REVIEW_LEAD_UNGATED = (
    'The file(s) below are the files for this issue. No fix has '
    'been applied yet.'
)
show = list(originals)
lead = REVIEW_LEAD_UNGATED
```

On a task where no edit applied, this call is the third request for edits, framed as a review. Both the opening clause and the set of exposed tasks change, so an outcome on a newly exposed task cannot be attributed to exposure alone, and the original-gate control of Appendix D compares the ungated arm with the ordinary keep arm on tasks where the original gate fires. The cross-model review arm of Appendix D.3 changes the client for the review call only. Draft and retry stay on the worker model and the templates are unchanged. The harness source also contains an older substitution interface that routes both retry and review to a substitute client. It is a different intervention from cross-model review, and no comparison in this paper pools the two.

The primary entrypoint requests at most 4 096 output tokens per call, with temperature zero and seed 1234. These settings specify decoding but do not guarantee identical responses across separate serves. Each arm normally reruns its prefix. Consequently, prompt equality, input-token equality, and a common seed do not by themselves establish equality of the pre-review text.

# B MEASUREMENT INVENTORY, EVIDENCE SOURCES AND COMPUTE

The studies use different task sets, workers and budgets because they answer different questions. This appendix identifies those settings, shows where observations are reused, and separates the resources charged to a policy from those spent on the wider campaign.

## B.1 UNITS, TASK SETS AND SERVING

A *task* is one benchmark instance. A *census* runs a specified model, harness, serving configuration and arm set over a fixed task manifest. A census can contain several arms and each arm can make several calls. Repeating a 60-task manifest adds task-replicate observations without adding unique tasks. Saved prefixes, continuation branches, evaluator repeats and model calls are distinct units throughout.

The historical primary slice E60 contains 60 SWE-bench Verified tasks selected by deterministic round-robin from the 11 repositories represented in its under-15-minute tier. H30 contains 30 tasks from the 15 to 60 minute tier. S59 removes the one E60 task that could not run in the optional-critic environment. The newer C7 cohort contains 200 previously unused Verified tasks and supports the controlled review studies. These cohort constructions do not license simple-random-sampling claims for the full benchmark. Live and ProMax cohorts are analyzed separately, and corrected task crosswalks link campaign identifiers to benchmark instances.

The historical harness uses greedy decoding with seed 1234 and a 4 096-token output cap. Table 4 records its serving configurations and those of the second harness. The new C7 same-draft and evidence studies use Qwen3-Coder-30B-A3B at 65 536 context tokens. Sequential control uses its own worker and action configuration in Appendix G. A result is not pooled across context limits, checkpoint revisions or worker identities merely because its policy label is the same.

Table 4: **Serving configurations of the later measurements.** The initial ladder was served earlier under a different inference-server version on a two-GPU node. Its fourth configuration, a community AWQ serving of the 235B model, was retired before any rerun and could not be re-measured, so the latest run carries no observation for it and none is reported. The official 235B-FP8 configuration appears only in the second-harness families. Qwen3-Coder-30B-A3B is the only model served at two context limits, which is why its 32k and 64k cells are never pooled.

| Later serving configuration | Context tokens | H100 GPUs / tensor parallel |
|---|---|---|
| Qwen2.5-Coder-7B-Instruct | 32 768 | 1 / 1 |
| Qwen2.5-Coder-32B-Instruct | 32 768 | 2 / 2 |
| Qwen3-Coder-30B-A3B-Instruct | 65 536 or matched 32 768 | 2 / 2 |
| Qwen3-235B-A22B-Instruct-2507-FP8 | 65 536 | 8 / 8 |
| Llama-3.1-8B-Instruct, bf16 mirror | 32 768 | 1 / 1 |
| Llama-3.3-70B-Instruct, bf16 mirror | 32 768 | 4 / 4 |

The Llama configurations use community bf16 mirrors with reconciled weight-digest receipts. The 235B official FP8 serving belongs to the second-harness studies, while the initial AWQ configuration remains retired. The execution manifests distinguish the primary harness, OpenHands optional critic, mini-SWE-agent observations and the later production source census. A static source census is not an execution of every configuration it classifies. The evaluator records identify the grader version for each outcome, while the new evaluator-repeat study measures scoring stability separately.

### B.2 Measurement families and observation reuse

The initial ladder is exploratory. The repeated primary study, its serving repair, the matched-context and cross-family comparisons, the harder tier, redrafting, cross-model review and ungated review form the historical measurement families. The serving repair replaces one mixed-instance replicate rather than adding a fourth. The matched-context comparison reuses the 32B censuses and adds 30B-A3B at 32k. The new precision reads extend specific cells and do not make the reused observations independent.

Table 5: **The studies behind the main text.** Counts are task or state counts, not model calls. Rows that reuse C7 or source controller tasks are not independent samples. Each contrast keeps its own paired unit, worker, budget and reference.

| Study | Worker or configuration | Unit and cohort | Role |
|---|---|---|---|
| Historical exposure and return | Qwen and Llama configurations in Table 4 | E60, three repeated censuses per primary comparison, plus H30 | Gate, token bill and rerun policy contrasts |
| 64k replication and pooling | Qwen3-Coder-30B-A3B | Three new E60 replicates, pooled with three repaired historical replicates | Stability of the rerun contrast |
| Same-draft review (E1) | Qwen3-Coder-30B-A3B, 64k | 200 C7 tasks, one saved state per task | Review and harmonized review versus retention |
| Trigger, allocation and evidence (E2, E3, E6) | The same 64k worker | The same 200 C7 tasks | Random trigger, capped alternatives and public-test feedback |
| Transfer and repetition (E4, E5) | The same 64k worker | 150 Live tasks, 60 ProMax tasks, 60 repeat-study tasks | Scope and sources of variation |
| Sequential confirmation | Qwen3.6-35B-A3B and API workers | 308 source entries from 305 issues, 250 new Live issues, 100 issues per API comparison | Repeated decisions under explicit budgets |
| Information acquisition | Qwen3.6-35B-A3B | 84 reveal/mask states on 84 canonical issues, 82 acquire/no-acquire states | Value of receiving and acquiring observations |
| Controller follow-up | Qwen3.6-35B-A3B | 308 source entries from 305 issues reused across arms | Calibration, ablation, cost and always-review comparison |

The second-harness studies retain their separate host and container conditions, with S59 for optional-critic measurements and E60 for observation feedback. Their corrected cell inventory and finish-critic extension appear in Appendix J. Stage-token accounting reuses call ledgers. Trace attribution reuses paired outcomes and draft bytes. Policy replays reuse recorded inputs and selected actions. The public HAL trace read launches no new model experiment. These operations cannot be summed into a total number of experimental tasks.

### B.3 Analysis rules and reconciliation

The historical repeated studies and alternatives at the trigger have author-controlled rules specifying their decisions and stop conditions. The 7B insufficient-exposure amendment, the serving repair and the adjudication refinements remain part of that analysis history. The initial census and the original E1 through E6 pilots are exploratory or retrospective. The completed new studies carry named decision, denominator, cohort, stopping and multiplicity rules in their registration ledger. A later reanalysis or pooled read does not inherit the status of an independent new experiment.

Task mappings, serving records, saved states, execution transcripts and policy inputs connect each outcome to the conditions under which it was obtained. These records support checks of the comparison itself, while the uncertainty analysis addresses what can be inferred from its sample. Tables

report the final estimates with their available uncertainty. Appendix E distinguishes pointwise intervals, exact tests, adjusted tests and equivalence claims.

### B.4 Resource accounts

The latest resource ledger separates allocated GPU capacity from measured utilization, and the cost ledger separates shared campaign cost from deployment cost charged to a policy. Table 6 reports the corrected campaign boundaries. Dollar entries are attributed charges under the exported pricing and allocation rule rather than provider invoices. Allocated GPU-hours multiply the number of reserved GPUs by the serving window and therefore do not measure device utilization.

Table 6: **Campaign resource accounts.** Every listed call joins its account. Capacity and amortized dollars are alternative cost accounts and must not be added. The new campaign's 820 GPU-hours comprise 640 deployment hours and 180 offline-evaluation hours. Their capacity charges are $2 528 and $711, respectively. n.a. means the summary provides no corresponding allocated-capacity total.

| Campaign | Joined calls | Allocated GPU-h | Capacity USD | Amortized USD |
|---|---|---|---|---|
| Historical | 4 602 | 210 | 829.50 | 149.15 |
| Earlier controller | 960 | n.a. | n.a. | 36.60 |
| Source controller | 16 473 | n.a. | n.a. | 510.73 |
| New review and control | 24 722 | 820 | 3 239.00 | 586.31 |

The cost-boundary table also reports 149.1 historical and 582.2 new-campaign utilization GPU-hours. It separately records controller computation and offline evaluation where applicable. For example, the source controller account lists $4.012 of controller computation and $265.25 of excluded offline evaluation. These columns describe their exported scopes and are not added to an episode charge without the corresponding allocation rule. The same physical prefix can be shared once in the campaign account while being charged to each policy that requires it in a deployment comparison.

This table is not a whole-project resource total. It does not aggregate the later follow-up campaign with every earlier exploratory serving window. Model tokens, tool time, capacity charges, utilization and deployment dollars answer different resource questions. The confirmation comparisons use reconstructed per-policy deployment charges. The original source and follow-up summaries retain different charge accounts, whose effect on comparison is stated in Appendix G. The review-tax comparison uses stage tokens without converting them into policy savings.

### B.5 External token traces

The HAL analysis reads the public trace corpus associated with the Holistic Agent Leaderboard (Kapoor et al., 2026). The reconciled corpus account retains 373 runs across eleven benchmark families and a median per-run input-token share of 95.0 %. Each run contributes its own input-token ratio to that median.

The 95.0 % value is a median of per-run ratios rather than a token-weighted aggregate. HAL corroborates the general prevalence of input-heavy calls. It does not identify review-stage expenditure, savings from removing review, uncached prefill work or any behavior's outcome benefit. Cache reuse, token pricing and serving configuration can change the monetary burden even at the same token ratio.

## C Formal statements behind the argument

This appendix states three propositions, one corollary and two remarks that the main text uses in words. Each follows from the definitions of Section 2 and Appendix C.1, the control target of Section 5, and the control flow of the harness. None uses an assumption that the paper does not already make, and none is an empirical result. Their purpose is to make explicit what the gate commits, what the removal design identifies, why correctness alone cannot decide the action after

an edit, and in what sense the gate and the state-blind schedule are restrictions of the control target. The first subsection collects the measured quantities that the statements use.

### C.1 Measured quantities and the all-task decomposition

Section 2 defines the keep arm, the strip arm for behavior $b$, the keep-fired set $T_b$ with $n_b$ of $N$ tasks, the exposure $e_b = n_b/N$ and the per-firing contrast $d_b$ of Equation 1. The all-task contrast $\Delta_b = \frac{1}{N}\sum_{w=1}^{N}(Y_w^K - Y_w^{-b})$ splits over $T_b$ and its complement, and therefore decomposes exactly as

$$\Delta_b = e_b d_b + R_b, \qquad R_b = \frac{1}{N}\sum_{w \notin T_b}(Y_w^K - Y_w^{-b}), \tag{4}$$

so the triggered term vanishes at zero exposure, where $d_b$ is undefined. We stratify on firing as Reddy et al. (2026) do and additionally retain the residual $R_b$ as a separate term. The residual can be nonzero because independently executed arms may already differ on tasks where review did not fire. Same-arm reruns and repeated evaluations distinguish trajectory variability from outcome-oracle variability (Appendix E).

### C.2 Exposure and the review share under the applied-edit rule

For task $w$, let $A_w \in \{0,1\}$ indicate that the draft applies at least one edit and let $B_w \in \{0,1\}$ indicate that the retry, when it runs, applies at least one edit. Let $p_1$ be the fraction of the $N$ tasks with $A_w = 1$ and $p_2$ the fraction of tasks with $A_w = 0$ that have $B_w = 1$, both empirical rates over the task set rather than population probabilities. Let $\bar{c}_D > 0$, $\bar{c}_T \geq 0$ and $\bar{c}_R > 0$ be the mean stage tokens of a draft call, of a fired retry call and of a fired review call.

**Proposition 1** (Exposure follows applicability)**.** *Under the three-call rule of Section 2:*

1. *retry fires exactly when $A_w = 0$ and review fires exactly when $A_w = 1$ or $(A_w, B_w) = (0,1)$, so the exposures are $e_{\text{retry}} = 1 - p_1$ and $e_{\text{review}} = p_1 + (1-p_1)p_2$, and a task rescued by retry is counted in both.*

2. *The mean keep-arm stage tokens per task are $\bar{c}_D + (1-p_1)\bar{c}_T + e_{\text{review}}\bar{c}_R$, and the review share*

$$s(p_1, p_2) = \frac{e_{\text{review}}\,\bar{c}_R}{\bar{c}_D + (1-p_1)\bar{c}_T + e_{\text{review}}\,\bar{c}_R} \tag{5}$$

   *is nondecreasing in $p_1$ and in $p_2$ at fixed per-firing costs, and strictly increasing in $p_1$ whenever $\bar{c}_T > 0$ or $p_2 < 1$.*

3. *Review-stage tokens per task, $e_{\text{review}}\bar{c}_R$, are nondecreasing in both rates, whereas total tokens per task need not be, since their derivative in $p_1$ is $(1-p_2)\bar{c}_R - \bar{c}_T$.*

*Proof.* Part 1 restates the control flow, since the harness retries only when nothing applied and reviews whenever an edit exists after the draft or the retry, and taking fractions over tasks gives the two exposures. For part 2, write $s = X/(M+X)$ with $X = e_{\text{review}}\bar{c}_R$ and $M = \bar{c}_D + (1-p_1)\bar{c}_T$. The partial derivatives of $e_{\text{review}}$ are $1-p_2$ in $p_1$ and $1-p_1$ in $p_2$, both nonnegative, so $X$ is nondecreasing in both rates, while $M$ is nonincreasing in $p_1$ and constant in $p_2$. The map $X \mapsto X/(M+X)$ is increasing for $M > 0$ and the map $M \mapsto X/(M+X)$ is nonincreasing for $X \geq 0$, which gives the monotonicity. Strictness in $p_1$ follows when $p_2 < 1$, since then $X$ strictly increases and $\bar{c}_R > 0$, or when $\bar{c}_T > 0$, since then $M$ strictly decreases and $X > 0$ because $p_2 = 1$ forces $e_{\text{review}} = 1$. Part 3 is immediate from the expression for the total. □

The proposition is the formal content of the gate finding in Section 3. A model whose drafts apply more often raises $p_1$, which lowers retry exposure and raises review exposure at the same time, and the review share of tokens rises with it even when every per-firing cost is unchanged. The per-firing cost $\bar{c}_R$ enters separately, and Section 3 shows that its level is dominated by prompt input, so a longer prompt raises the share through $\bar{c}_R$ while a stronger editor raises it through $p_1$. The stage identity $C_b/N = e_b\bar{c}_b$ of Section 2 separates the two channels exactly in the matched-context record of Table 8. Review tokens per task-replicate rise from 3 220 to 3 800, a factor of 1.180, while exposure rises by $167/137 = 1.219$ and tokens per firing fall by a factor of 0.968. Their product

is the expenditure ratio, up to rounding. Increased exposure therefore more than offsets the slightly smaller review calls in this comparison. Input dominates each call, but neither input nor output length increases with the model change. The proposition holds per-firing costs fixed, whereas the empirical decomposition allows them to differ.

For the removal comparison, the keep arm produces a draft $D_w^K$ after its draft and retry calls and then reviews it, and the strip arm produces its own draft $D_w^S$ and stops. Let $Y_w(D, 1)$ and $Y_w(D, 0)$ denote the outcome of draft $D$ with and without the review call, so that $Y_w^K = Y_w(D_w^K, 1)$ and $Y_w^{-b} = Y_w(D_w^S, 0)$. Outcomes are binary.

**Proposition 2** (Same-draft effect and divergence term)**.** *For every fired task,*

$$Y_w^K - Y_w^{-b} = \tau_w + \delta_w, \qquad \tau_w = Y_w(D_w^K, 1) - Y_w(D_w^K, 0), \qquad \delta_w = Y_w(D_w^K, 0) - Y_w(D_w^S, 0), \tag{6}$$

*where $\tau_w$ is the effect of reviewing the draft that was actually reviewed and $\delta_w$ is the difference between the two unreviewed drafts, and the per-firing contrast of Equation 1 satisfies $d_b = \bar{\tau} + \bar{\delta}$ with both averages taken over $T_b$. Consequently:*

1. *If $D_w^K = D_w^S$ and the benchmark evaluation is a deterministic function of the final patch, then $\delta_w = 0$.*
2. *On a discordant pair exactly one of $\tau_w$ and $\delta_w$ is nonzero: either review changed the outcome of the keep draft, or the two unreviewed drafts already had different outcomes, never both.*
3. *When $\delta_w = 0$, the observed difference equals $\tau_w$. Identical drafts certify this condition under a deterministic evaluator. For different drafts, the two recorded endpoint outcomes alone do not generally identify $\tau_w$.*
4. *$d_b$ is an unbiased estimate of the mean same-draft effect on $T_b$ when the conditional expectation of $\delta_w$ given keep-arm firing is zero. Identically distributed drafts in the two arms are not sufficient for this, because $T_b$ is selected on the keep arm's draft. Under the stable-evaluator assumption, it holds when drafts are deterministic given the task, and when drafts are exchangeable between arms and gate membership is the same event in both arms.*

*Proof.* Equation 6 adds and subtracts $Y_w(D_w^K, 0)$, and averaging over $T_b$ gives $d_b = \bar{\tau} + \bar{\delta}$. Part 1: with equal drafts the two subtracted terms are evaluations of the same patch, which coincide under deterministic evaluation. Part 2: $\tau_w$ and $\delta_w$ each lie in $\{-1, 0, 1\}$, and a discordant pair has $\tau_w + \delta_w \in \{-1, 1\}$, which forces one term to be zero and the other to be $\pm 1$. Part 3 follows from parts 1 and 2: the observed difference equals $\tau_w$ when $\delta_w = 0$, while a nonzero $\delta_w$ reproduces the observed sign with $\tau_w = 0$. Distinguishing these cases requires the unreviewed outcome $Y_w(D_w^K, 0)$ or evidence that determines it. Under deterministic evaluation, an identical strip-arm draft supplies that outcome. Part 4: taking expectations gives $\mathbb{E}[d_b] = \mathbb{E}[\bar{\tau}] + \mathbb{E}[\bar{\delta} \mid \text{keep fired}]$. Under stable evaluation, if drafts are deterministic given the task, the two drafts coincide and $\delta_w = 0$ by part 1. If the drafts are independent draws from one per-task distribution, the keep-fired event depends on $D_w^K$ but not on $D_w^S$, so $\mathbb{E}[Y_w(D_w^K, 0) \mid \text{keep fired}]$ and $\mathbb{E}[Y_w(D_w^S, 0)]$ differ in general, and they coincide when firing is the same event in both arms, since then the selection acts identically on both drafts. □

**Corollary 1** (Attribution in the repaired study)**.** *In the repaired 30B-A3B study, 19 of the 20 discordant pairs on the review-fired set have different pre-review patch bytes. Their observed contrast cannot be assigned to the same-draft review effect without observing the corresponding unreviewed candidate. The remaining pair has an identical pre-review patch and a regression. Under a stable evaluator, only that pair directly isolates review within this rerun comparison.*

The result concerns identification, not whether a changed patch can help or harm. Review may change a patch even when the two arms had already diverged. The shared-patch study instead fixes the candidate before branching and directly evaluates both continuations (Section 4). Repeated evaluations of an identical patch address the stability assumption separately.

### C.3 CORRECTNESS ALONE DOES NOT DECIDE THE ACTION

Let the cost weight satisfy $\lambda \geq 0$. For an observable state $x$, let $q(x) = \mathbb{E}[Y^{\text{stop}} \mid x]$ be the probability that the retained patch resolves the task, and for an action $a$ that can change the patch, such as review with revision, let $\rho_a(x)$ be the probability that the action followed by the continuation $\pi_0$ resolves the task given $x$ and an incorrect retained patch, and $\eta_a(x)$ the probability that it leaves the task unresolved given $x$ and a correct retained patch.

**Proposition 3** (Value of a patch-changing action)**.** *Under the control target of Equation 3,*

$$\tau_a(x) = \rho_a(x)\,\big(1 - q(x)\big) - \eta_a(x)\,q(x). \tag{7}$$

*Consequently:*

1. $\tau_a(x) \leq 0$ *wherever* $q(x) = 1$*, so an action on a patch known to be correct never satisfies* $\tau_a(x) - \lambda\kappa_a(x) > 0$ *when* $\kappa_a(x) \geq 0$*.*
2. *At* $q(x) = 0$ *the criterion holds if and only if* $\rho_a(x) > \lambda\kappa_a(x)$*, so an incorrect patch justifies acting only when its repair probability covers the weighted cost.*
3. *A positive probability that the retained patch is incorrect is necessary but not sufficient for positive incremental value, when added cost is nonnegative.*
4. *If* $\rho_a$*,* $\eta_a$ *and* $\kappa_a$ *are constant across states and* $\rho_a + \eta_a > 0$*, the criterion is equivalent to the threshold* $q(x) < (\rho_a - \lambda\kappa_a)/(\rho_a + \eta_a)$ *on the correctness probability, and it is never met when* $\rho_a \leq \lambda\kappa_a$*. If these quantities vary with* $x$*, a threshold on* $q(x)$ *alone need not reproduce the sign of* $\tau_a(x) - \lambda\kappa_a(x)$*. In particular, it cannot do so when two states with the same* $q$ *have opposite signs.*

*Proof.* Condition on whether the retained patch is correct, an event with conditional probability $q(x)$ given $x$. On the incorrect branch $Y^{\text{stop}} = 0$ and $\mathbb{E}[Y^{a,\pi_0}] = \rho_a(x)$, and on the correct branch $Y^{\text{stop}} = 1$ and $\mathbb{E}[Y^{a,\pi_0}] = 1 - \eta_a(x)$. Mixing with weights $1 - q(x)$ and $q(x)$ gives Equation 7. Parts 1 and 2 substitute $q = 1$ and $q = 0$, and part 3 combines them. For part 4, with constants the criterion $\tau_a(x) - \lambda\kappa_a > 0$ is affine and decreasing in $q(x)$, so it is a threshold on $q$, and the threshold is nonpositive when $\rho_a \leq \lambda\kappa_a$. If $\rho_a + \eta_a = 0$, the gain is zero and no positive-cost action is preferred. With state-dependent quantities, two states can share $q$ while having opposite signs of the criterion. Such a pair is a counterexample to any rule based only on $q$. Variation alone does not force such a pair to exist. □

The decomposition explains why predicting correctness and predicting an action's benefit are different tasks. Two equally likely-to-be-correct patches can have different repair opportunities, damage risks and costs. It does not imply that a learned gain policy must outperform a correctness gate empirically. The additional source repetition in Section 5 does not resolve that comparison. A predicted damage filter likewise supplies no certified population harm bound.

*Remark* 1 (The gate and the schedule as restrictions of the control target). Let $F$ be the event that an edit applies. The applied-edit gate is the constant policy that selects review at every state in $F$, and a state-blind schedule is a policy whose action depends on the state only through the step index, the remaining budget and the set of feasible actions. Both are therefore restrictions of the estimator class of Equation 3: the gate corresponds to a constant positive estimate of $\tau_a - \lambda\kappa_a$ for review, and the schedule to an estimate that ignores the patch and its evidence. Two consequences follow without proof. A one-step selector that chooses, at a state in $F$, an action maximizing $\tau_a(x) - \lambda\kappa_a(x)$ over a feasible set that contains review and stopping can do no worse than choosing review followed by the same continuation $\pi_0$, because the maximum over a set containing review is at least the value at review. This comparison does not establish a guarantee for repeatedly executing the selector with a different continuation. Also, no evidence about the patch, including a test output that has already been paid for, changes a gate that always selects review. Neither consequence says that the gate is bad. They say that the gate's return on $F$ is whatever the mean incremental value of review happens to be, which Section 4 measures under specific prompts and contexts, and that a policy conditioning on the state and on disclosed evidence is the natural comparison, which Section 5 records.

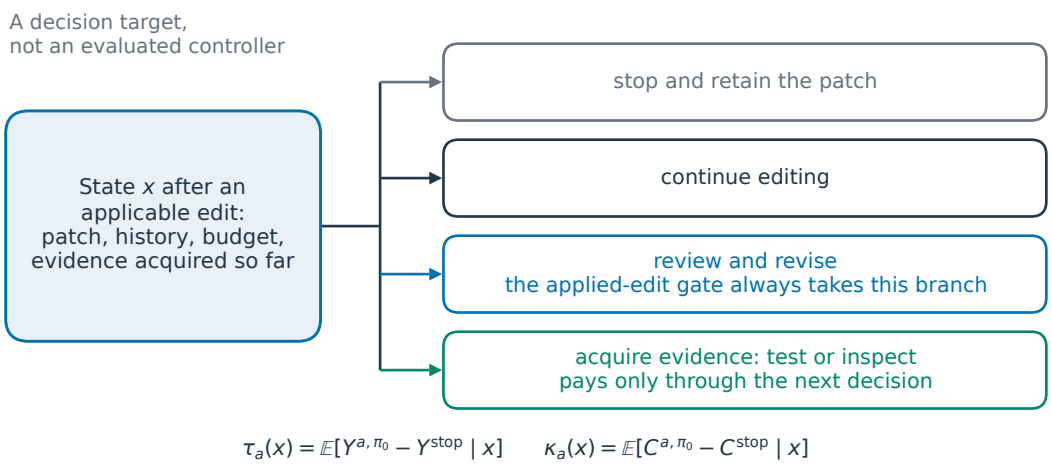


Figure 5: **The decision that an applicable edit opens.** The applied-edit gate takes the review branch in every state. A controller would instead compare actions by their incremental value $\tau_a(x)$ and cost $\kappa_a(x)$ from Equation 3. Acquired evidence has value only through the decision that follows it. The figure depicts a decision target. The deployed controller uses stochastic score-based selection, whose replay is reported separately.

Figure 5 draws the decision that an applicable edit opens. The gate takes the review branch at every state, whereas the control target compares the available actions by their incremental value and cost. The figure depicts this decision target rather than a guarantee for an estimated or stochastic policy.

*Remark* 2 (Selection consistency under the specified rule). Let $s_a(x)$ be the logged score of action $a$ at a decision state $x$ whose feasible set contains stopping. If the selected action is $a^\star(x) \in \arg\max_a s_a(x)$, for a greedy implementation of the objective in Equation 3, then $s_{a^\star(x)}(x) \geq s_{\text{stop}}(x)$ at every decision. By contraposition, a decision at which the selected action scores below stopping was not produced by that rule applied to the logged scores: either the logged scores are not the inputs of the deployed rule, or the deployed rule is not the specified maximization, for instance a stochastic rule that samples from recorded probabilities.

This remark applies to greedy maximization. The deployed controller instead samples from a softmax of risk-filtered gain-minus-cost scores. Selecting an action below the stop score is possible under that stochastic rule and is not a replay failure. The supplied frozen heads, feature vectors, action order and random draws reproduce its logged choices (Appendix G). This verifies the selection computation on the supplied representations, not calibration of its gain estimates or optimality of its decisions.

## D Exposure, expenditure and repeated policy comparisons

This appendix reports the trusted task-level firing records and stage accounts underlying Sections 3 and 4. Replicates repeat the same tasks. The matched 32k, repaired 64k and harder-tier settings are reported separately.

### D.1 Exposure on fixed task slices

The matched Qwen configurations have 137 and 167 review firings over 180 task-replicates. The Llama configurations have 55 and 101. The harder-tier counts are 39 and 49 over 60 task-replicates. All three comparisons have higher review exposure under the higher-applicability configuration. These are descriptive comparisons of fixed configurations, with no claim that architecture or parameter count alone causes the difference. The gate identity is a statement about the source-defined control flow (Proposition 1), while the table reports the directly logged review events.

Table 7: **Exposure and resolution are distinct quantities.** Slash-separated values are separate replicates on the same tasks. The harder tier has two replicates on 30 tasks. No replicate creates a new independent task.

| Configuration | Tasks / replicate | Review firings | Review-arm solves |
|---|---|---|---|
| Qwen2.5-Coder-32B, 32k | 60 | 45/46/46 | 17/23/22 |
| Qwen3-Coder-30B-A3B, 32k | 60 | 55/56/56 | 20/21/18 |
| Qwen3-Coder-30B-A3B, 64k | 60 | 53/53/54 | 21/26/26 |
| Llama-3.1-8B, 32k | 60 | 18/19/18 | 20/23/21 |
| Llama-3.3-70B, 32k | 60 | 34/33/34 | 22/24/14 |
| 32B, harder tier | 30 | 19/20 | 11/9 |
| 30B-A3B, harder tier | 30 | 24/25 | 13/8 |

Table 8: **Review-stage expenditure.** Each row covers 180 task-replicates. Per-task values divide input plus output tokens by 180, and shares divide by all model tokens in the same review arm. The 32k Qwen rows are the matched comparison.

| Configuration | Calls | Input tokens | Output tokens | Tokens / task | Share (%) |
|---|---|---|---|---|---|
| 32B, 32k | 137 | 482 881 | 96 758 | 3 220 | 20.78 |
| 30B-A3B, 32k | 167 | 572 896 | 111 127 | 3 800 | 23.72 |
| 30B-A3B, 64k | 160 | 554 000 | 111 461 | 3 697 | 23.43 |
| Llama-8B, 32k | 55 | 198 353 | 35 278 | 1 298 | 8.33 |
| Llama-70B, 32k | 101 | 352 813 | 67 557 | 2 335 | 14.81 |

### D.2 Stage-token accounts

For the matched Qwen rows, the all-stage denominators are 2 789 193 and 2 884 051 tokens. Review input plus output totals are 579 639 and 684 023, giving 20.78 % and 23.72 %. The corresponding review input fractions are 83.31 % and 83.75 %, and the mean review outputs are 706.3 and 665.4 tokens. All 137 and 167 review calls join to their firing records. The review returns `DONE` on 64 and 69 calls and applies edits on 64 and 85 calls. These event counts have different meanings and are not complementary outcomes.

At the repaired 64k setting, draft, retry and review account for 2 098 777, 75 910 and 665 461 tokens. Their sum is 2 840 148, with shares 73.90 %, 2.67 % and 23.43 %. This account separates the cost of the review stage from the other computation in the harness.

For a review arm with total tokens $C_K = P_K + C_R$ and a no-review arm with $C_{-R} = P_{-R}$,

$$C_K - C_{-R} = C_R + (P_K - P_{-R}).$$

Thus removing review saves exactly the review-stage tokens only when the remaining calls cost the same. Separately regenerated prefixes need not satisfy that condition. Similarly, equal context limits do not equalize tokenizers, hardware usage or prices. The matched stage's recorded uncached and cached-price accounts are $2.010 and $1.083 for 32B and $2.205 and $1.197 for 30B-A3B. These quantify the specified price treatment, not realized invoice savings from a controlled cache intervention.

### D.3 Precision extensions and alternative allocations

The precision extensions add repeated executions to the same task slices. Their task-cluster intervals preserve this dependence. A cross-model reviewer changes the review model, while redrafting uses the extra generation opportunity for another candidate. Neither contrast identifies a universal advantage for one use of computation.

**Redraft selection and the budget rule.** The redraft arm replaces review at the same applied-edit trigger with a prompted second draft from the original files, using the generation limit of the replaced

call. If the second attempt applies an edit, its patch is selected. Otherwise, the arm retains the first post-retry patch. This is a prompted alternative, with no selection based on benchmark outcomes. The trusted selection check confirms that evaluation does not precede patch selection. Appendix A gives the prompt.

The historical budget rule compares whole-arm token use. Let $C_w^K$, $C_w^{-R}$ and $C_w^{\text{redraft}}$ denote total input plus output tokens for the review, no-review and redraft arms on task-replicate $w$. On the review-fired set $T_R$, its tolerance is

$$\left|\text{median}_{w\in T_R}(C_w^K - C_w^{\text{redraft}})\right| \le 0.25\left|\text{median}_{w\in T_R}(C_w^K - C_w^{-R})\right|.$$

The denominator is the median whole-arm difference after removing review, not review's gross stage expenditure. The rule therefore does not assert equal realized tokens or equal monetary cost. It defines the historical comparison protocol. The precision-extension table below reports the trusted outcome contrast without treating the earlier budget medians as a newly verified cost result.

Table 9: **Extended policy contrasts.** Values follow the treatment direction recorded for each comparison. These are additional repetitions combined with the respective original cells, not new task populations. Each supplied task-cluster interval includes zero.

| Comparison | Fired pairs | Net favorable | Contrast (pp) | 95 % interval |
|---|---|---|---|---|
| 32k re-serve | 278 | +7 | +2.52 | [−1.88, 7.10] |
| Llama-3.1-8B | 94 | −2 | −2.13 | [−7.36, 2.36] |
| Llama-3.3-70B | 170 | +6 | +3.53 | [−2.62, 8.01] |
| Cross-model reviewer | 200 | +9 | +4.50 | [−0.44, 10.88] |
| Redraft | 268 | +8 | +2.99 | [−1.90, 7.34] |

The historical 7B removal read remains excluded for insufficient exposure: its governing record has only one review firing, so it cannot estimate a useful review effect. The original ungated-review controls also retain their stop-rule status. Where the control changed outcomes on tasks already eligible for ordinary review, the rule stopped interpretation of the no-patch extension. Those stopped cells are not evidence that an additional turn on a failed-to-edit task helps or is harmless. The withdrawn 0.56 noise threshold is not reinstated.

The historical ungated-review study remains a boundary case: when no edit exists, its extra call is a further repair attempt with a different prompt (Appendix A). It cannot establish the return of reviewing an existing patch. The controlled shared-patch and execution-feedback studies instead intervene after the candidate is fixed (Appendix F). No extrapolation to all 500 SWE-bench Verified tasks is reported as a measured result. The full-set extension remains unfilled in the trusted result package.

## E VALIDITY: DEPENDENCE, SERVING REPAIR AND ATTRIBUTION

A paired outcome difference can be measured precisely without identifying the effect of reviewing one patch. This appendix explains the assumptions behind the uncertainty calculations, the variability introduced by rerunning generation and evaluation, and the evidence needed to attribute an outcome change to review.

### E.1 THRESHOLDS, INTERVALS AND REPEATED TASKS

A *read* is an estimate under a specified analysis rule, not necessarily a new experiment. A *trio* contains three replicates, and a pooled read can reuse the tasks of an earlier read. For independent paired differences $D_w = Y_w^K - Y_w^{-b}$ with zero mean and discordance probability $f$, $\text{Var}(d_b) = f/n_b$. The corresponding normal-approximation critical threshold and minimum detectable effect at power $1-\beta$ are

$$t_\alpha = z_{1-\alpha/2}\sqrt{f/n_b}, \qquad \delta_\beta = \left(z_{1-\alpha/2} + z_{1-\beta}\right)\sqrt{f/n_b}. \tag{8}$$

A critical threshold, a minimum detectable effect, a confidence interval and a practically justified equivalence margin answer different questions. None can substitute for the others. The historical

sign-flip analyses additionally assume independent discordant signs across task-replicate observations. Repeated tasks can violate that assumption, which is why Table 1 reports the supplied task-cluster intervals for the updated reads.

The repaired historical 64k comparison has 6 salvages and 14 regressions over 160 firings, giving $-5.0$ points. Its exact paired $p$ is $0.1153$, and its task-cluster interval is $[-10.2, 0.7]$. The new trio has 17 salvages and five regressions over 162 firings, giving $+7.4$ points $[1.8, 13.2]$ and exact $p = 0.0169$. Their pooled contrast is $+1.2$ points $[-3.0, 5.3]$ with exact $p = 0.6440$. The new trio and the pooled read are not two independent confirmations. The sign change across trios is evidence against treating one rerun estimate as a stable local effect.

The other historical comparisons also estimate policy differences. At 32B, 1 salvage and no regressions over 137 firings give $+0.7$ points. Llama-3.1-8B has 2 salvages and three regressions over 55 firings, giving $-1.8$ points. Llama-3.3-70B has 2 salvages and five regressions over 101 firings, giving $-3.0$ points. Their supplied task-cluster intervals are $[-4.3, 5.7]$, $[-6.8, 3.2]$ and $[-8.0, 2.0]$, respectively. None resolves a directional benefit under its exact comparison. These estimates pool the paired counts across replicates, rather than averaging the rounded replicate contrasts.

The precision extensions likewise leave the corresponding policy effects unresolved. The 32k 30B-A3B re-serve gives $+2.5$ points $[-1.9, 7.1]$ over 278 firings. The extended Llama-8B and Llama-70B reads give $-2.1$ $[-7.4, 2.4]$ over 94 and $+3.5$ $[-2.6, 8.0]$ over 170. Cross-model review gives $+4.5$ $[-0.4, 10.9]$ over 200, and redrafting gives $+3.0$ $[-1.9, 7.3]$ over 268. These extensions increase precision within their stated settings without establishing a general benefit or harm of review.

### E.2 Noise floors and multiplicity

The new 64k same-arm check observes three flips in 60 pairs, a 5.0 % rate with Wilson interval $[1.7, 13.7]$. The matched 32k checks and the governing historical reference each record 19 flips in 180 pairs, or 10.6 %. These are configuration-specific observations. A same-arm floor from one stratum cannot be assigned to another without an assumption about their variability. In particular, review not firing does not by itself establish that two prefixes and serving conditions were identical.

For independent Bernoulli draws with common success probability $p$, expected disagreement satisfies

$$2p(1-p) \le \tfrac{1}{2}. \tag{9}$$

An observed finite-sample rate above one half is possible, and dependent draws need not obey the model. The inequality is a diagnostic of a noise model rather than an automatic validity test. The earlier 0.56 threshold was withdrawn and is not used in this draft.

The export includes Holm-adjusted exact tests for its named families. We distinguish those decisions from pointwise task-cluster intervals. E2 illustrates why the distinction matters: its pointwise interval excludes zero but its adjusted exact test does not resolve the contrast. The original-prompt E1 row does not supply an adjusted value, so its positive interval and exact $p$ support the named comparison, not a familywise claim covering every prompt, benchmark and allocation. No result establishes a deployment risk guarantee.

### E.3 Complete outcomes and operable reference arms

A model-dispatch failure is a missing execution, not an observed failed test. Assigned unavailable evaluations retain their assigned denominator under the operational endpoint and are distinguished from observed failures. The evaluator-check patch set includes recovery of eight formerly unknown terminal evaluations from the earlier campaign. This recovery is scoped to that patch set and does not remove the unavailable endpoints retained in the later source policy comparison. The new E1, E2, E3, E4 and E6 outcome comparisons have no unknown evaluations. The sequential comparisons retain the unavailable endpoints stated in Appendix G.

If every no-review outcome on the firing set fails, the paired contrast reduces to

$$Y_w^{-b} = 0 \text{ for all } w \in T_b \quad \Longrightarrow \quad G_b = 0, \qquad d_b = \frac{1}{n_b} \sum_{w \in T_b} Y_w^K. \tag{10}$$

The comparison then demonstrates the reference arm's failure under that contract and supplies no balanced estimate of the two directions of change. The historical context-removal cell is flagged this way and supplies no review-effect estimate. Before reading any other cell, the fired-set counts must reproduce its paired contrast, the all-task decomposition must retain the off-trigger residual, and the call sequence must satisfy the gate contract. The firing check reports no set-equality failures across 4 740 records. These checks establish arithmetic and execution consistency rather than the completeness of the benchmark oracle.

### E.4 Limits of the outcome oracle

Resolution means passing the repository evaluator, not a proof of semantic correctness. Test coverage, ambiguous specifications and benchmark familiarity can affect that endpoint. Pairing on task identity controls which issue each arm attempts, but it does not ensure that the tests distinguish every correct and incorrect patch. The evaluator-repeat study in Appendix F separately measures the stability of scoring an identical patch and does not estimate test completeness or memorization.

The exposure and review-tax findings count firings and tokens, so they do not depend on the correctness oracle. Every resolution gain in Sections 4 and 5 is conditional on the stated benchmark evaluator. A positive effect on that endpoint remains informative within that scope without establishing correctness on untested behaviors.

### E.5 Serving repair and historical sensitivity

The historical 30B-A3B comparison originally included a replicate split across serving instances after an interruption. Replacing that replicate on one instance is a serving repair, not an additional independent replicate. The serving records verify that the replacement used one instance, and the repaired comparison is the historical reference used in the paper. Before repair, 6 salvages and 12 regressions give $-3.75$ points over 160 firings. The repaired count of 6 and 14 gives $-5.0$ points. Both remain policy contrasts because a single serving instance does not force separate executions to share a draft.

The earlier sensitivity over eight disputed regression pairs concerned their attribution, not whether the eight paired differences existed. Using the pooled counts, excluding those contributions while retaining the denominator moves the contrast from $-8/160 = -5.0$ points to zero. This is a sensitivity to attribution under a fixed denominator, not a newly fitted test, a gate-membership analysis or an estimate of a same-draft intervention. The byte comparison below now addresses the underlying attribution question directly.

The matched-context re-serve controls the configured context ceiling only. Architecture, training, effective context use and remaining serving differences remain bundled with model identity. Its sign difference from the 64k historical estimate, with overlapping intervals, does not establish a causal effect of context length.

### E.6 Byte-level attribution in the repaired comparison

The paired records contain 21 outcome-discordant pairs across all tasks in the repaired 64k cell. Restricting to the 160 review-fired pairs leaves the 20 discordances behind the reported contrast. Nineteen have different draft bytes and one has identical draft bytes. The second-reader adjudications agree with these classes. The all-task record and the firing-set record therefore have different denominators and are not conflicting estimates.

Different drafts do not show that review did nothing. Review can visibly change one patch even when the no-review arm starts from another patch. Equation 2 explains why the resulting final-outcome difference then combines two possible sources. The same-draft regression supplies one identified adverse example, while the new 200-task fork identifies the average local effect in its own cohort. Neither observation substitutes for the other, and their different task and execution conditions preclude pooling them as one controlled effect.

Table 10: **Most paired changes precede the opportunity to isolate review.** Counts join the repaired historical cell's paired outcomes to its firing membership. Different drafts prevent identifying an individual same-draft review effect from that pair alone. The single byte-identical pair is a regression. The table uses the full discordant set among reviewed tasks.

| Draft relation on the review-firing set | Salvages | Regressions | Total |
|---|---|---|---|
| Different draft bytes | 6 | 13 | 19 |
| Identical draft bytes | 0 | 1 | 1 |
| All discordant review-fired pairs | 6 | 14 | 20 |

# F Follow-up studies E1 through E6: fixed drafts, allocation and evidence

The six studies separate questions that a review-removal contrast cannot answer by itself. E1 estimates the effect on a fixed candidate. E2 varies the trigger, E3 allocates a capped turn, E4 changes the task cohort, E5 measures repetition at different stages, and E6 changes the evidence available during review. The latest completed studies use Qwen3-Coder-30B-A3B at a 65 536-token context limit. E1, E2, E3 and E6 share the C7 cohort of 200 previously unused SWE-bench Verified tasks, so their results are related measurements rather than four independent replications. E1, E3 and E6 use separate executions and saved states, so an arm named review need not have the same outcome count across studies. E4 has separate Live and ProMax cohorts.

## F.1 Evidence, endpoints and uncertainty

Each comparison links the assigned task to its saved state, executed branches, final evaluation and expenditure. The original E1 through E6 pilots are retrospective. The later studies specify their decision rules, cohorts and denominators in a registration ledger, whose public timing has not been independently verified. We therefore report estimates and uncertainty with their stated analysis rules, and distinguish adjusted tests where available.

Resolution is the benchmark evaluator's binary outcome. Every assigned outcome is available in the new E1, E2, E3, E4 and E6 comparisons. E1 has one saved prefix per task. Contrasts pair the treatment and reference on the same task, and the supplied intervals use task clusters. The tables report the supplied estimates and intervals, rounded to one decimal percentage point. We independently reconstruct the counts, means and E1 branch pairing, while retaining the supplied intervals without reconstructing their original resampling draws. Repeated prefixes or continuations remain repeated measurements of their source tasks. Pointwise intervals are not simultaneous confidence statements across all studies.

## F.2 E1: same-draft review and prompt compatibility

The 200 saved states each have an applicable edit and are forked into retention, original review, harmonized review and a sham. All 200 fork records verify byte identity at the branch root. The sham issues the review request but discards its edits, and its outcome agrees with retention on all 200 tasks. The original prompt retains the historical edit-only system instruction while allowing an unchanged response in the user message. The harmonized prompt allows that response consistently. It changes the instruction compatibility while retaining the candidate and worker.

Retention resolves 76 of 200 tasks. Original review resolves 86, with 12 repairs and two regressions, and the harmonized prompt resolves 81, with seven repairs and two regressions. Table 1B gives their paired contrasts. The original prompt's exact paired $p$ is 0.0129. The harmonized comparison's exact $p$ is 0.1797 and its recorded adjusted value is 0.3594. The original-prompt result supports a positive local effect on this cohort. It neither proves that instruction incompatibility is beneficial nor establishes a difference between the two review prompts, since those are different hypotheses. Retention consumes no review call, so this is a local efficacy comparison at unequal cost.

### F.3 E2 AND E3: WHEN TO REVIEW AND HOW TO SPEND THE TURN

E2 compares a quality-based invocation rule with random invocation under a 16 000-token cap. Their resolution rates are 44.0 and 40.5 %. Although the supplied pointwise interval excludes zero, the exact paired $p = 0.0654$ and the adjusted $p = 0.1963$ do not resolve the comparison. The inference depends on the test, so we retain the unresolved decision. Mean realized token use is 5 144.5 versus 4 071.2, with a reported gate cost of 210 tokens and no cap violations. The two policies share a cap but do not realize the same expenditure.

E3 compares original review with structured review, redrafting, resampling and a longer initial solve. Each arm resolves 39.0 % of the 200 tasks. Each paired contrast is zero with interval $[-2.0, 2.0]$ points, bounding the aggregate advantage at the supplied precision while allowing individual gains and losses. Mean realized tokens are 11 651.7 for original review, 11 542.4 for structured review, 11 619.8 for redrafting, 11 506.3 for resampling and 11 403.8 for the longer solve. All satisfy the 16 000-token cap. This comparison supports no outcome advantage for the alternative allocations on this cohort. Equal caps and nearby means do not by themselves establish matched realized costs.

Table 11: **Trigger, allocation, transfer and review evidence.** Contrasts and supplied task-cluster intervals are in percentage points. All studies use Qwen3-Coder-30B-A3B at 64k. E2, E3 and E6 reuse the 200-task C7 cohort and are not independent replications of E1. E4 uses its named benchmark cohorts. E2 and E3 have a 16 000-token cap, with no recorded cap violations. All outcomes are available.

| Comparison | Tasks | Contrast (pp) | 95% interval (pp) |
|---|---|---|---|
| **E2. Invocation rule** | | | |
| Quality trigger minus random | 200 | $+3.5$ | $[+0.3, +6.7]$ |
| **E3. Allocation of the capped turn** | | | |
| Structured review minus original | 200 | $+0.0$ | $[-2.0, +2.0]$ |
| Redraft minus original review | 200 | $+0.0$ | $[-2.0, +2.0]$ |
| Resampling minus original review | 200 | $+0.0$ | $[-2.0, +2.0]$ |
| Longer solve minus original review | 200 | $+0.0$ | $[-2.0, +2.0]$ |
| **E4. Transfer cohorts** | | | |
| Review minus retain, Live | 150 | $+1.3$ | $[-7.4, +10.0]$ |
| Review minus retain, ProMax | 60 | $+1.7$ | $[-13.4, +16.8]$ |
| **E6. Evidence during review** | | | |
| Execution minus exploration | 200 | $+9.0$ | $[+4.1, +13.9]$ |
| Exploration minus fixed context | 200 | $+1.5$ | $[-1.4, +4.4]$ |
| Fixed context minus retain | 200 | $+0.5$ | $[-1.7, +2.7]$ |

### F.4 E4: TRANSFER OF THE LOCAL CONTRAST

The Live cohort contains 150 tasks and the ProMax cohort (Shi et al., 2026) contains 60. On Live, review resolves 88 tasks against 86 under retention. On ProMax the counts are 30 and 29. Their intervals span both material benefit and harm (Table 11), so the original-prompt benefit on C7 does not establish the same benefit on these cohorts. The task identifiers map to official benchmark instances, but the selected cohorts are not probability samples of their benchmark populations and are analyzed separately.

### F.5 E5: GENERATION, CONTINUATION AND EVALUATION VARIATION

The nested repeat study records 51 disagreements in 120 full-draft pairs, a rate of 42.5 % with supplied Wilson interval $[34.0, 51.4]$. It records zero disagreements in 240 frozen-prefix continuation pairs and zero in 240 identical-patch evaluator pairs. Each zero count has a supplied upper interval endpoint of 1.6 %. The repeated pairs arise from 60 source tasks and therefore do not supply 240 independent tasks. These are descriptive pair-level intervals rather than task-cluster guarantees. The

contrast between levels localizes observed variability to full generation in this configuration, while a zero count does not prove a deterministic continuation or evaluator.

The separate evaluator check re-evaluates 219 terminal patches from the earlier controller campaign and finds three changed outcomes, 1.37 % with Wilson interval $[0.47, 3.95]$. It also completes the eight formerly unknown evaluations. This check establishes a small observed evaluator disagreement rate in that patch set, distinct from the new study's zero disagreements. It does not warrant imposing one universal noise floor on all model and serving configurations.

### F.6 E6: EXECUTABLE EVIDENCE CHANGES THE REVIEW INTERVENTION

E6 compares retention, fixed-context review, review with repository exploration and review with public-test execution plus feedback. The 200-task execution comparison passes its functional sentinel. Public tests execute on all 200 assigned tasks and the resulting feedback reaches the reviser on all 200. The execution-minus-exploration gain is $+9.0$ points $[4.1, 13.9]$. Exploration alone gains $+1.5$ $[-1.4, 4.4]$ over fixed-context review, and fixed-context review gains $+0.5$ $[-1.7, 2.7]$ over retention. Thus the strongest effect is associated with executable evidence, whose delivery the transcript and sentinel controls now verify.

The execution arm uses a mean of 4 540.5 model tokens and 58 tool-wall seconds in the supplied account. The execution branch loses resolution on two tasks that the retain branch resolves, so a positive net gain does not mean every revision helps. The export does not provide false-approval or false-rejection rates for this comparison, and we make no decision-accuracy claim from resolution alone. Separate candidate-bank diagnostics contain 600 saved-candidate evaluations on 60 issues in the earlier bank, with 240 resolved outcomes, and 200 candidates on the C7 issues in the new bank, with 92 resolved outcomes. These counts describe the evaluated candidate sets. They do not identify a repair effect, and the new bank's reuse of C7 prevents treating it as an independent replication of the paired execution study.

### F.7 CONSEQUENCES FOR POST-EDIT CONTROL

The studies establish a positive local review effect in one controlled comparison, unresolved trigger and transfer effects, no aggregate gain from the tested alternative allocations, substantial pre-review repetition variation, and a benefit from public-test feedback. These are different decision targets. A controller must still choose whether a specific candidate merits another action, whether acquiring evidence is worth its cost, and when to stop. Section 5 evaluates that sequential problem with its own worker, task cohort and full deployment charges.

## G SEQUENTIAL CONTROL: IMPLEMENTATION, OUTCOMES AND EVIDENCE USE

This appendix separates three kinds of evidence: the primary sequential policy comparison and its confirmations, the controlled disclosure and acquisition comparisons, and follow-up continuations on the same tasks. The comparisons use executed trajectories, their outcomes and costs, and the frozen policy artifacts. The follow-up continuations add useful operating points, but their checkpoint and cost boundaries differ from those of the original policy episodes.

### G.1 TASKS, POLICIES AND THE DEPLOYED RULE

The source task manifest contains 308 assigned entries from 305 canonical SWE-bench-Live issues. The crosswalk agrees with the canonical instance field for all 588 pilot, training, development, calibration and test entries. The respective split sizes are 40, 120, 80, 40 and 308. The additional confirmation cohort contains 250 canonical issues. API comparisons use 100 issues per worker. Qwen3.6-35B-A3B is the local source worker and remains the reviewer when the worker and reviser change to Gemini 3.8 Flash, Grok 4.6 or Grok 4.5. The supplied API artifact records provider builds for all three workers. The task crosswalk and serving identifiers are retained in the reproducibility record.

The source policy comparison has seven arms, each with the same six available action types, a maximum of six decisions and a \$0.45 primary episode cap. Stop retains the current patch. Continued editing and the two review actions may revise it. Inspection and public testing acquire observations without changing the patch. Table 12 states the implemented controller and reference rules. The always-review arm is a separate follow-up continuation, rather than an eighth arm of the original common-prefix comparison.

Table 12: **The deployed controller is stochastic.** A sampled action need not maximize its score. The myopic rule is the supplied deterministic gain-only variant. The always-review row is evaluated in follow-up continuations.

| Policy | Decision rule |
|---|---|
| Learned | Linear gain, cost and predicted-damage heads over recorded state-action features. Samples from the softmax of $(\widehat{\tau}_a - 4\widehat{\kappa}_a)/0.25$ over feasible actions with predicted damage at most $0.05$. A logged uniform draw and canonical action order determine the choice |
| State-blind schedule | Action probabilities use the step, remaining budget and feasibility. Patch and evidence features are excluded. The fitted schedule need not place positive probability on every feasible action |
| Correctness gate | Predicts current-patch correctness from observable signals, then maps that prediction to an action using a fixed rule |
| Fixed cadence | Follows a periodic action schedule |
| Myopic | Selects the largest predicted gain with cost weight zero and a canonical tie rule |
| Worker continuation | Continues the coding worker without the learned controller |
| Test-all | Acquires a public test whenever eligible, then makes the downstream continuation decision |
| Always review | Selects review at an applicable checkpoint, subject to the episode's stopping and budget constraints |

The frozen feature contract has ten state channels, six action indicators and eight representation channels. The state channels describe the decision index, remaining budget, whether a patch exists, patch churn, the last public-test result, the number of tests and reviews, the last review's completion marker, whether the patch is newer than the test, and budget consumed. The three heads are linear in the 24-dimensional vector. The representation channels are supplied for each recorded state-action pair. Their encoder is not included, so the artifact reproduces decisions from the recorded features but does not independently construct those features for an unseen repository state.

### G.2 Replay and prediction diagnostics

Running the supplied policy on its recorded inputs reconstructs 48 251 head-value rows and 43 151 propensity rows across 8 323 scored decisions. Every selected action reproduces. This includes 5 669 source-worker decisions, 573, 618 and 613 decisions for the three API workers, and 850 myopic decisions. The source-worker count spans the primary study, confirmations and acquisition extensions, so it is not a count of independent test tasks. All 168 disclosure first decisions also reproduce from the separately supplied scores. Later disclosure decisions without recorded feature vectors are outside that replay check. Because the policy samples actions, choosing one with a score below stopping is consistent with its rule.

Replay verifies computation conditional on the supplied feature vectors. It does not establish that a score estimates the target in Equation 3. The development diagnostics contain 960 action branches from 160 saved states on 80 issues. The reported Brier score of $0.1228$ and AUC of $0.9018$ compare the gain head with terminal resolution, rather than with the action's incremental outcome relative to stopping. In particular, the stop branch has predicted gains between $0.02$ and $0.98$, whereas its true increment relative to itself is zero. These diagnostics therefore assess prediction of a terminal label and do not validate incremental-gain calibration.

Predicted and realized branch costs correlate strongly, with Pearson $r = 0.9854$, Spearman $\rho = 0.9534$, and respective means \$0.0782 and \$0.0777. A deterministic score-ranking diagnostic selects a best observed branch at 85 of 160 states, compared with 95 for always stopping on this branch bank. This diagnostic uses a score maximizer rather than the deployed stochastic choices, and does not establish ranking superiority over simple references. Its six branches per state are not independent test observations or a population regret guarantee. Descriptive action frequencies also differ after passing and failing tests: the learned policy stops at 42.6 % of the 94 decisions following a pass and 29.0 % of the 31 following a failure. Those conditional frequencies mix different visited states. Controlled disclosure, rather than this association, supplies the intervention evidence below.

**Observed repair and damage along policy trajectories.** The checkpoint evaluations distinguish action effects on initially failed and initially successful patches. Under learned control, continued editing repairs 21 of 434 failed patches and damages two of 256 successful ones. Execution review repairs six of 138 and damages one of 84. Fixed-context review repairs 68 of 675 and damages nine of 370, giving a net $59/1\,045 = 5.65$ percentage points across its visited states. Under the schedule, fixed-context review repairs 11 of 801 and damages 15 of 460, a net $-4/1\,261 = -0.32$ points. These different policies visit different states, and repeated checkpoints are not independent tasks. The contrast is therefore descriptive evidence about where actions were used, not a matched-state causal comparison of those actions or a certificate for the predicted-risk filter.

### G.3 Primary policy outcomes and uncertainty

For an operational resolution rate, each assigned task entry receives one unit of weight. An unavailable terminal evaluation earns no solve credit but is not treated as an observed failed test. Three pairs of task entries share a canonical issue, so uncertainty resamples the 305 canonical issues with all entries and paired arms kept together. We use 10 000 percentile bootstrap resamples with seed 20260922 and recompute the ratio of successful entries to assigned entries in every resample. This preserves the 308-entry estimand while accounting for canonical-task dependence. Intervals are pointwise and describe the supplied comparisons. Completion bounds separately assign unavailable endpoints their most favorable and least favorable outcomes. They are sensitivity ranges, not confidence intervals.

The primary learned and state-blind arms resolve 143 and 123 of 308 entries. There are 22 favorable and two adverse paired discordances, giving $+6.49$ points with interval $[3.57, 9.74]$. Both arms have three unavailable terminal evaluations, and arbitrary completion of these labels gives a sample difference between $5.52$ and $7.47$ points. Thus missing endpoint labels do not reverse this sample's ordering.

The other original source arms resolve 122 entries for the correctness gate, 118 for fixed cadence, 128 for myopic control, 119 for worker continuation and 126 for test-all. These outcomes share the source episodes and must not be counted as separate confirmations. Test-all acquires a test at the first decision in 294 episodes, compared with 71 for the learned policy, which tests in 126 episodes overall. The learned policy's 17 additional solves show that more testing does not reproduce its source outcome. The contrast also changes the continuation policy and does not isolate the value of acquisition. The repeat below does not reproduce superiority over test-all or the correctness gate.

### G.4 Confirmation, budget variation and workers

The source repeat and the additional cohort are separate executions from the primary source comparison. At the \$0.45 cap, the repeat resolves 123 versus 111 entries, $+3.90$ points $[0.97, 7.12]$, at mean deployment charges of \$0.1247 and \$0.1094. The additional cohort resolves 106 versus 90 of 250 issues, $+6.40$ points $[2.40, 10.40]$, at \$0.1211 and \$0.1073. The paired branches share their recorded initial state and evaluation within each comparison.

Budget variation spans caps of \$0.225, \$0.45 and \$0.90. The learned policy's source-repeat gains over the schedule are $3.25$, $3.90$ and $4.55$ points. On the additional cohort they are $5.20$, $6.40$ and $7.20$ points. Table 13 gives the intervals and actual charges, which remain well below the caps. A larger cap permits more computation without requiring it, so this is budget sensitivity rather than equal-spending evaluation.

Table 13: **The schedule advantage recurs, while gains over stronger references remain unresolved.** Charges come from the confirmation episode ledger. Intervals are pointwise 95 % paired canonical-issue bootstrap intervals. Source tasks comprise 308 entries from 305 issues.

| Reference | Cap ($) | Solves | | Difference (pp) | Mean charge ($) | |
|---|---|---|---|---|---|---|
| | | Learned | Reference | [95% interval] | Learned | Reference |
| **A. Source repeat versus state-blind schedule** ($n = 308$) | | | | | | |
| Schedule | 0.225 | 121 | 111 | +3.25 [0.33, 6.21] | 0.1119 | 0.1061 |
| Schedule | 0.450 | 123 | 111 | +3.90 [0.97, 7.12] | 0.1247 | 0.1094 |
| Schedule | 0.900 | 125 | 111 | +4.55 [1.30, 7.79] | 0.1263 | 0.1054 |
| **B. Additional cohort versus state-blind schedule** ($n = 250$) | | | | | | |
| Schedule | 0.225 | 103 | 90 | +5.20 [1.60, 9.20] | 0.1118 | 0.1028 |
| Schedule | 0.450 | 106 | 90 | +6.40 [2.40, 10.40] | 0.1211 | 0.1073 |
| Schedule | 0.900 | 108 | 90 | +7.20 [3.20, 11.60] | 0.1247 | 0.1079 |
| **C. Source repeat versus stronger references** ($n = 308$) | | | | | | |
| Correctness gate | 0.450 | 123 | 125 | −0.65 [-4.21, 2.91] | 0.1247 | 0.0732 |
| Test-all | 0.450 | 123 | 122 | +0.32 [-3.26, 3.90] | 0.1247 | 0.1304 |

Table 14: **Worker comparisons at the $0.45 cap.** Each row contains 100 paired issues. The learned policy improves on its schedule at higher mean charges but does not separate from the correctness gate. The reviewer remains Qwen3.6 across workers.

| Comparison | Solves | | Difference (pp) | Mean charge ($) | |
|---|---|---|---|---|---|
| | Learned | Reference | [95% interval] | Learned | Reference |
| **A. Gemini 3.8 Flash** | | | | | |
| Schedule | 54 | 43 | +11.0 [2.0, 20.0] | 0.0614 | 0.0458 |
| Correctness gate | 54 | 55 | −1.0 [-10.0, 8.0] | 0.0614 | 0.0378 |
| **B. Grok 4.6** | | | | | |
| Schedule | 52 | 41 | +11.0 [2.0, 20.0] | 0.0939 | 0.0702 |
| Correctness gate | 52 | 52 | +0.0 [-10.0, 10.0] | 0.0939 | 0.0519 |
| **C. Grok 4.5** | | | | | |
| Schedule | 48 | 38 | +10.0 [2.0, 18.0] | 0.0812 | 0.0638 |
| Correctness gate | 48 | 49 | −1.0 [-10.0, 8.0] | 0.0812 | 0.0455 |

Against the source repeat's correctness gate, the learned policy resolves two fewer entries and spends more: 123 versus 125, $-0.65$ points. Against test-all it resolves one more, 123 versus 122, $+0.32$ points. Both intervals include zero. The correctness gate's mean charge is \$0.0732, below the learned policy's \$0.1247, while test-all's is \$0.1304. The evidence therefore supports an advantage over the schedule without a reproduced advantage over every stronger reference.

Under the three pinned API workers, the learned policy resolves 54, 52 and 48 of 100 issues, compared with 43, 41 and 38 for the respective schedules. Each learned arm costs more than its schedule. Their correctness gates resolve 55, 52 and 49 issues at lower mean charges, and the learned-minus-correctness intervals include zero. These comparisons support recurrence of the schedule contrast under different workers. They do not establish that a learned incremental-value estimate is preferable to a correctness gate in those workers.

### G.5 Using an observation and deciding to acquire it

The disclosure comparison contains 84 paired states on 84 canonical issues. Both continuations start after the same public test has been acquired, both receive a \$0.09 acquisition charge, and neither may reacquire that test. The supplied inputs include the actual test payload for the reveal arm and the masked input for its partner. Four isolation checks per pair pass in all 336 cases, supporting that the test output is the information changed between the branches. The recorded first-decision scores and feature vectors reproduce all 168 selections.

Reveal resolves 47 states and mask resolves 34. Fourteen paired discordances favor reveal and one favors mask, giving $+15.48$ points with interval $[7.14, 23.81]$. One reveal endpoint is unavailable. Completing that endpoint gives a sample difference between $15.48$ and $16.67$ points. The mean recorded continuation-charge difference is $-\$0.0021$. The charge records include controller computation. Comparing these charges with the broader policy campaign still requires the accounting boundaries described below.

The first action changes in 34 of 84 pairs, or 40.5 %. All these changes replace inspection in the masked arm with editing or review in the reveal arm. This is direct evidence that disclosure changes decisions. Conditioning outcome comparisons on the changed-action subset would select on a post-treatment variable, so it would not identify how much of the total resolution effect flows through those transitions. A follow-up forced-action diagnostic resolves 10 of 33 selected states against one under masking, a 27.27-point difference $[9.09, 45.45]$. This result is conditional on that selected population, rather than an additional unconditional estimate of disclosure or its mediation.

The separate acquisition comparison includes 82 states and contrasts acquiring and using the test with not acquiring it. The respective outcomes are 47 and 33 solves, a 17.07-point difference $[7.32, 26.83]$, while the acquisition branch costs \$0.0913 more on average. This comparison adds acquisition cost to the decision. It does not mean that every test is worth acquiring, and it should not be combined with the 84-state disclosure contrast as though the two populations and references were identical.

### G.6 Follow-up continuations and resource sensitivity

The follow-up studies use the same 308 task entries, but their recorded initial checkpoints differ from those of the primary policy comparison. For 307 entries, the follow-up identifier points to a source checkpoint at step one rather than the original step-zero state. The remaining identifier does not join to a source state. Consequently, comparisons with the original learned and schedule arms are paired by task but not by initial checkpoint. This affects the interpretation of masking, always review, cost-weight variation and training-seed contrasts. This checkpoint difference limits the policy contrast.

The evidence-masked continuation resolves 126 entries at mean charge \$0.1207. Relative to the primary policy's 143, the paired difference is $5.52$ points $[2.90, 8.47]$. Its advantage over the original schedule is only three solves instead of twenty, an 85 % attenuation in this sample. Because initial checkpoints also differ, this contrast cannot isolate the effect of removing evidence features. The controlled disclosure comparison provides the stronger claim about evidence use.

The always-review continuation resolves 130 entries at mean charge \$0.1413. The task-paired primary-minus-follow-up difference is $4.22$ points $[1.63, 6.84]$. Varying the cost weight gives 138 solves at \$0.1099 for $\lambda = 8$ and 146 at \$0.1399 for $\lambda = 2$. The selected near-charge point gives 137 solves at \$0.1053, compared with 123 for the original schedule, $+4.55$ points $[2.29, 7.12]$. The values describe useful observed operating points, but the changed checkpoints and cost boundary prevent a common-prefix efficiency or frontier claim.

Two alternative fit seeds give 145 and 142 solves. Together with the primary fit's 143, their task-weighted gains over the original schedule are 7.14, 6.49 and 6.17 points, with a mean of 6.60 and a sample standard deviation of 0.50 points. Two repetitions on a 100-task subset give 41 solves each for learned control and 35 each for the schedule. These outcomes document sensitivity within the supplied studies. The seed comparisons inherit the changed-checkpoint boundary, and reuse of the same tasks does not increase the number of independent issues.

### G.7 What the charge accounts establish

The primary comparison has two reported charge accounts. Summing the episode allocations gives mean charges of \$0.15045 for learned control and \$0.12050 for its schedule. The follow-up summaries report \$0.12302 and \$0.10747 for those same primary arms. Each account orders learned control above the schedule, so the resolution gain is accompanied by higher spending under either account. The records used here do not reconcile the two totals to one common deployment boundary. We therefore identify which account a display uses and do not equate either with an independently verified invoice.

For the newly supplied follow-up continuations, the per-episode full-charge fields reproduce their summary means. The always-review mean of \$0.14135 lies above the follow-up summary's learned mean but below the original episode account's learned mean. Its cost ordering therefore depends on the chosen boundary, even before accounting for the changed initial checkpoint. A claim of verified savings over always review is not supported. Likewise, the near-charge point is close to \$0.10747 but is not exactly equal to it.

The confirmation-study episode charges reproduce their reported means, and all evaluated budget levels have no recorded cap violations. Shared physical work must be counted once in a campaign total and charged to every deployed policy that would require it. Offline benchmark evaluation is outside the deployment decision, while controller computation and observations required by the policy belong inside it. The supplied cost receipts improve coverage of those categories, but their availability alone does not resolve every allocation boundary between the primary and follow-up accounts.

## H Earlier controller comparison

The earlier campaign provides context for the larger sequential evaluation. The trusted export links 960 final evaluations to 60 tasks per policy in each of two worker configurations. Table 15 reports these completed outcomes. The timing analysis reuses the same episodes and supplies no additional independent sample.

The learned policy has three more solves than always review and the correctness gate in each configuration, and two more than the myopic policy. Such small counts motivate the stronger references, larger samples and explicit policy replay of Section 5. The campaign used 345 allocated GPU-hours. Campaign capacity and policy deployment charges are distinct accounts.

**Risk filtering is not a harm certificate.** A predicted-risk filter does not establish a population harm guarantee. For $m$ fixed candidates, $n$ independent calibration units and failure probability $\delta$, a Hoeffding union bound at zero observed harm has radius $\sqrt{\log(m/\delta)/(2n)}$. With $m = 5$, $\delta = 0.05$ and $n = 40$, this is 0.2399. Achieving a bound of 0.05 under this particular conservative argument requires at least 922 zero-harm units. This calculation is a limitation of that certificate, not a universal sample-size requirement. The current policy comparisons report observed outcomes without claiming a certified five-percent damage limit.

Table 15: **Earlier sequential policy outcomes.** These are descriptive counts from the reconciled campaign. They are not pooled with the larger source comparison or counted as an independent replication of it.

| Policy | 32B solves / 60 | 30B-A3B solves / 60 |
|---|---|---|
| Learned | 23 | 31 |
| Myopic | 21 | 29 |
| Correctness gate | 20 | 28 |
| Always review | 20 | 28 |
| Never review | 18 | 26 |
| Random | 19 | 27 |
| Fixed cadence | 19 | 27 |
| Worker continuation | 20 | 28 |

# I Extended related work and comparison scope

This appendix answers which measurement designs are closest to ours and where each differs in intervention, resource reporting and repeated-run evidence. The relevant literature asks whether review improves an existing answer, how a harness selects that opportunity, which outcome contrast an intervention measures, and what computation it consumes. Table 16 compares the closest designs. Most report resource use in some form, and same-arm variability is reported by only some of them, in forms that measure different quantities. Among the designs in Table 16, none combines a deterministic edit-triggered gate with stage-level input and output token accounting and same-arm reruns. These distinctions position the two historical findings. At fixed cost per review call, an applied-edit gate spends more on review as edit applicability rises (Section 3, Proposition 1). Separate reruns can differ before review, so we use shared-prefix forks to estimate review's local effect (Section 4, Proposition 2).

## I.1 Self-correction depends on feedback and the comparison

Intrinsic self-correction and externally informed repair answer different questions. Huang et al. (2024) find that models struggle to correct reasoning without external feedback and sometimes degrade their answers. For code generation, Olausson et al. (2024) find variable repair gains across models and budgets, with stronger results when feedback improves. Jin & Chen (2026) study requirement-conformance judgments on paired canonical and buggy function-level programs, finding false rejections that vary with the requested explanations and corrections. Their prompt comparison motivates measuring damage to correct candidates, although it does not establish the effect of review on repository repair.

Activation itself can conceal a weak intervention. Long et al. (2026) distinguish frequent confirmatory rechecks from corrective ones within reasoning traces. Mirzaei (2026) compares self-inspection with repeated sampling at measured generated-token cost and identifies a Reflexion configuration that never activates on the smallest tested model. Their mathematics tasks, output-token budget, and majority-vote selection differ from our coding protocol. Verma (2026) compares blind resampling, a content-free failure notice, execution feedback, and reflection in function-level synthesis. Their repair arms share an initial program and matched attempt limits, while actual tokens vary. These studies make the alternative call and its selection rule part of the scientific comparison.

Execution feedback is not unconditionally helpful either. On SWE-bench Verified, Tao et al. (2026) hold a repair agent fixed and find that tests written by the untrained base test-writing agent lower its resolution below a no-test baseline, whereas tests from a stronger model raise it. Lin et al. (2026) prohibit code execution in three program-repair agents and find resolution gaps that are small and not statistically significant, averaging 1.25 percentage points on the two commercial agents. The token saving is 56 to 62 percent for Claude Code on both benchmarks and varies widely for the other two agents. Zhang et al. (2026a) vary only the returned observation in a matched agent loop and find, in question answering and fact verification, that persistent misleading feedback can leave

a tool-using agent worse off than no feedback at all. These results tie the value of feedback to its reliability, and our historical review call receives no execution feedback.

Positive review results identify conditions beyond our feedback-free call. SWE-Review combines repository exploration, executable checks, and revision, and its agentic reviewer outperforms single-turn fixed-context review in decision accuracy and in resolve rate after revision (Wang et al., 2026a). SWE-Search combines search and iterative feedback (Antoniades et al., 2025). Xiang et al. (2026) report asymmetric cross-model review results without test execution, and Anonymous (2026) train small critics to guide larger coding agents. Reviewer identity and available evidence therefore matter independently of generator size. OpenCodeReview adds bounded context tools and a separate comment-filtering reflection stage (Li et al., 2026a). Its review-comment metrics and token costs concern the quality of feedback, rather than executed patch-repair outcomes.

### I.2 Harness and model must be evaluated together

Source inspection identifies the mechanisms available to an agent, and Rombaut (2026) characterizes thirteen pinned coding scaffolds this way. Runtime studies then measure how configurations change outcomes: Sghaier et al. (2026) evaluate sequential harness releases with a fixed model, and Lewis (2026) compare two configurations across context windows and transfer the frozen treatment to additional models. Component interactions require finer comparisons, which Liu (2026) studies through a factorial design over scaffold components on reasoning and question-answering tasks.

These findings do not imply that weaker models always benefit more from scaffolding. eun Cho (2026) reports opposing harness-complexity effects for the evaluated frontier chat and reasoning models. Fan et al. (2026) likewise find that the effect of a harness component differs across the evaluated models. In business-oriented tasks, Yang et al. (2026) find that adaptation works best when small models have sufficient base capability. Li et al. (2026b) show that the sign of a gating signal's association with the value of extra computation can reverse across environments and model backbones. Our 64k and 32k estimates for one model and task set differ in sign with broadly overlapping intervals, which is consistent with such a reversal and does not establish one.

Production review already selects on more than an applied edit. RADAR gates semantic diff review on eligibility rules and risk scores and validates deterministically (Adams et al., 2026). It lies outside the ten-harness source census of Appendix J, and its observational efficiency comparisons do not estimate the return of a review call. It therefore motivates the post-edit policy comparisons of Section 5 without informing their estimates.

Context and edit interfaces require similarly precise comparisons. Repository instruction files studied by Gloaguen et al. (2026) and Khatri (2026) differ from the target-file contents supplied by our oracle-localized harness. Our paths-only arm in Appendix D removes those contents, so any contrast from it would concern oracle-supplied file contents rather than autonomous localization or instruction-file value, and the latest run carries no outcome for that arm. Chen et al. (2026b) instead vary prompts to change test-writing behavior, so their outcome comparisons concern that intervention. Aider's edit-format documentation explicitly distinguishes model compatibility with whole-file and search/replace output (Aider, 2026). It motivates testing whether entry into an applied-edit gate reflects format compatibility as well as problem-solving ability. This paper reports no interface comparison, because the records of the planned interface diagnostic are aggregate summaries without task-level executions. By construction, the gate's firing set coincides with the set of applicable edits (Proposition 1).

### I.3 Estimands, activation, and repeated-run variability

Reddy et al. (2026) relate aggregate lift to firing frequency and firing-stratum outcomes, report non-firing results, and remove gates individually. Kaliyev & Maryanskyy (2026) restrict paired evaluation to tasks where coordination is active under both compared protocols and measure an empty-store noise envelope. Our decomposition (Equation 4) keeps the keep-fired term and the off-trigger residual as separate terms of one identity, so a discrepancy outside the fired set stays visible instead of being absorbed into an aggregate lift, and the structural checks of Appendix E require that residual to be retained.

The conditioning event also fixes the population being compared, and this is where the closest prior work and our formal statement meet. Principal stratification defines comparison groups through the joint potential values of an intermediate variable under both interventions (Frangakis & Rubin, 2002). In our design the intermediate variable is the draft, and Proposition 2 states the consequence: a separately executed removal contrast on the fired set is the sum of the effect of reviewing the draft that was actually reviewed and the difference between two unreviewed drafts, and identical drafts make the second term vanish under a stable evaluator. Different drafts can also have equal unreviewed outcomes, but separate reruns do not observe both outcomes needed to establish that equality. The replay literature reaches the same point experimentally. Causal Agent Replay reconstructs the intervention state and samples continuations under step interventions (Shah, 2026), the Replay Gap compares model-switch branches with same-model controls after reconstructing a shared prefix (Gonuguntla, 2026), and in multi-agent debugging Luan et al. (2026) find that generic feedback-based repair shows no advantage over plain rerunning at an equal attempt budget, and attribute a repair to an intervention only after replaying the recorded execution up to the intervention point. A whole-policy comparison and the effect of reviewing one fixed draft are therefore different estimands even with matched task identifiers. Corollary 1 applies this to the repaired study (Appendix E.5). There, 19 of the 20 discordant pairs on the review-fired set have different pre-review patch bytes. The remaining pair shares its draft and is a regression, so only that pair directly isolates review under a stable evaluator. The shared-patch fork (Section 4) is the design that sets the divergence term to zero by construction.

Repeated runs address an additional uncertainty. Sam-Bodden (2026) reports six changed task outcomes between two same-protocol runs on seventy tasks in an oracle-localized coding setup. Gonuguntla (2026) finds that same-model control divergence depends on the serving configuration. In a clinical protocol, Bellibatlu et al. (2026) measure per-action instability under repeated identical conditions at provider-default sampling. Chen et al. (2026a) separately measure rerun variability and prompt-perturbation sensitivity for single-turn tool calls, reporting call counts and Gemini thinking tokens. These studies measure different quantities, including action divergence, ever-flip fractions, and paired outcome disagreement. Their numerical floors cannot be transferred to our tasks and serving configurations, which require their own same-arm measurements and prefix checks. The paper's own measurements of this kind are the pooled same-arm flip rate over non-triggered task-replicate observations and the serving-instance comparison of Appendix E, the same-arm floors recorded with the alternative trials at the trigger, and the repetition study E5 of Appendix F.

### I.4 Stage expenditure and alternative uses of compute

Stage-level token accounting exists for development-phase pipelines, where Tokenomics maps ChatDev calls to phases and separates input, output and reasoning tokens (Salim et al., 2026), and for generation, review and revision stages across generators (Wang et al., 2026a). At the whole-harness level, Vats & Golev (2026) report up to a forty-fold difference in tokens per solved task while paired pass-rate intervals include zero except for the largest gap. Our stage totals differ in what they condition on: they localize expenditure to a gate entry condition and include the calls whose response makes no further edit.

Gross stage expenditure differs from the cost of an alternative policy. Cost-of-Pass combines monetary expenditure and solution success (Erol et al., 2026), while Weinberger & Hozez (2026) show that caching and changed trajectories can offset local token reductions. Adaptive allocation depends on prompt difficulty (Snell et al., 2025), and Effective Feedback Compute distinguishes retained useful information from raw expenditure (Zhang et al., 2026b). Together with the resampling comparisons above, these studies motivate measuring end-to-end alternatives. Our input-plus-output totals identify a review burden, while an allocation policy requires matched costs and outcome measurements.

### I.5 Controller learning and the remaining comparison

Adaptive computation is a close predecessor of the controller comparisons. Damani et al. (2025) learn reward distributions conditional on an input and computation budget, Zhai et al. (2026) train a lightweight predictor of cost-sensitive oracle allocations, and Ong et al. (2025) route between models from preference data. On mathematics problems, Dip et al. (2026) train recoverability gates that

Table 16: **Close comparisons differ in intervention and measurement target.** The table covers the removal, replay and accounting designs closest to the historical study of Sections 3 to 4. The controller comparisons of Section 5 are positioned in the text against LoopArena and LearnStop instead. *Not reported* means that the reviewed methods and results did not report a same-arm variability measurement. It does not establish that no such data exist. Repeating a policy, measuring direct disagreement, and sampling counterfactual continuations answer different questions.

| Study | Intervention or measured process | Resource reporting | Repeat-run evidence |
|---|---|---|---|
| Reddy et al. (2026) | Deterministic action gates, firing and non-firing strata, individual removals | No extra model calls from gate predicates | Disjoint-seed replication on the same tasks |
| Kaliyev & Maryanskyy (2026) | Paired coordination protocols, activity-conditioned subsets | Protocol and trial counts | Configuration-equivalent noise envelope |
| Shah (2026) | Reconstructed states and stochastic step interventions | Bounded continuation budget | Repeated counterfactual continuations |
| Gonuguntla (2026) | Model-switch branches and same-model control forks | Rollout and branch-pair counts, with per-step tokens recorded but not analyzed | Control divergence by serving configuration |
| Verma (2026) | Shared initial programs, blind retries and feedback variants | Attempt limits and output tokens | Greedy retry duplication documented |
| SWE-Review (Wang et al., 2026a) | Fixed candidate PRs, agentic review, revision and selection alternatives | Stage tokens and sample budgets | Not reported |
| Lewis (2026) | Paired harness configurations, context windows and model transfers | Prompt tokens, model turns and solver wall time | Not reported |
| Sam-Bodden (2026) | Oracle-localized context representations | Context tokens per resolved issue | Two same-protocol runs on seventy tasks |
| Chen et al. (2026a) | Frozen tool-call reruns and prompt perturbations | Call counts and Gemini thinking tokens | Rerun and perturbation variability |
| Tokenomics (Salim et al., 2026) | Development-phase activity in ChatDev | Phase shares and token types | Not reported |
| This study | Behavior removals, activation and retained-prefix inspection | Gross stage input and output tokens | Same-arm reruns and serving audit |

decide whether a verification call on a frozen solver's answer is worth its tokens, and find that a larger initial reasoning budget reaches similar accuracy with fewer tokens. The learned policy of Section 5 is therefore compared against a state-blind schedule and a correctness gate rather than credited with the principle of selective computation. Proposition 3 distinguishes correctness prediction from action value: an incorrect patch may be unrepairable, and an intervention may damage a correct one. Constant repair, damage and cost terms yield a correctness threshold, but state-dependent terms need not. This distinction motivates comparing the policies and does not predict that learned control must win. The additional source repetition leaves its contrast with the correctness gate unresolved (Section 5).

LoopArena holds a coding worker fixed while evaluating controller decisions, restored-state alternatives, task slices and complete tasks, and its no-control and fixed-control references motivate equivalent baselines for repeated editing (Wang et al., 2026c). LearnStop compares a learned stopper against scalar exit rules under a shared calibration protocol, charges the answer probes used to make the decision, and finds that the winner depends on the task regime (Dong et al., 2026). Yi & Song (2026) train a harness controller offline that learns when to check before submitting. It outperforms both a forced-check schedule and behavior cloning in five of eight evaluated settings, with outcome gains concentrated in its coding domain and two adapted benchmarks. HarnessBridge trains a small controller that compresses observations and rejects proposed actions, and transfers it from smaller generators to larger commercial models (Wang et al., 2026b). AgentStop learns when to abort a local agent run from low-cost execution signals such as token log probabilities (Pham et al., 2026), and EET uses structured experience from earlier issue-resolution runs to end patch generation and selection early, reducing the cost of three software-engineering agents on SWE-bench Verified (Guo et al., 2026). These works make controller complexity, independent calibration and observation overhead necessary comparison axes.

Two works pose the acquisition decision directly. Ding et al. (2026) frame whether to run a test or retrieve context as a cost-uncertainty decision made by the acting model under a calibrated prior. Papamarkou et al. (2026) cast critic and verifier invocation as cost-sensitive sequential hypothesis testing and show, in retrospective replay over nine coding benchmarks, that belief-driven verification beats always-verify and single-critic gates when verification is expensive and the critics are informative but imperfect. Our comparisons build on controller separation (Wang et al., 2026c), shared-prefix replay (Shah, 2026; Gonuguntla, 2026; Luan et al., 2026), value-based allocation (Damani et al., 2025; Zhai et al., 2026), learned checking and stopping (Yi & Song, 2026; Dong et al., 2026; Pham et al., 2026; Guo et al., 2026), and cross-worker transport (Wang et al., 2026b). We use these ideas to ask which information should guide computation after a coding agent has produced a patch. The state-blind schedule supplies a reference that sees the step, remaining budget and action feasibility, while the reveal/mask comparison holds acquisition fixed to measure the value of using the output. The learned policy improves resolution over the schedule, and revealing the output improves the subsequent continuation. Replay verifies the stochastic selections from their recorded features. These findings support particular policy and information comparisons, while leaving calibration of incremental gain and superiority over correctness gating unresolved. The schedule's action probabilities can also differ from the learned policy's, so their outcome contrast alone does not isolate the effect of access to patch information. Appendix G distinguishes these questions and states the cost boundary of each comparison.

A training corpus with complete common-state branches can label local outcome differences directly. Partially observed action logs instead require actual logging propensities and adequate action support. Doubly robust policy evaluation combines reward and logging-policy models (Dudík et al., 2011), but cannot identify an unobserved action from a deterministic gate with no support. Moreover, one-step evaluation under a fixed continuation does not replace execution of a learned policy that changes future states, which is why Section 5 evaluates the controller on the episodes it produces. Training the reviser, as in self-correction learning (Kumar et al., 2025), is a different intervention and must remain separate from learning only the controller.

# J PRODUCTION TRIGGERS AND A FUNCTIONING FINISH CRITIC

The primary harness isolates an applied-edit trigger. This appendix asks which triggers occur in the inspected sources and whether the broader lesson survives when a functioning critic evaluates an agent's finish claim.

## J.1 PRODUCTION SOURCE CENSUS

The reconciled source inventory contains 186 gates across ten harnesses: OpenHands, SWE-agent, Aider, DevinFree, AutoCodeRover, Moatless, Lemo, Agentless, MarsCode and Codex CLI. Table 17 uses the final reconciled class of each gate. The four previously unassigned entries are classified as failure triggers, so all 186 enter this count. These are source-code classifications, not measured invocation frequencies or estimates of deployment prevalence.

Table 17: **Reconciled source inventory.** The successful-intermediate-event category contains five gates, none marked default-on in the trusted inventory. A trigger class alone does not determine the cost or value of the action it schedules.

| Trigger class | Gates | Enabled by default |
|---|---|---|
| Failure | 82 | 82 |
| Self-reported completion | 26 | 0 |
| Budget | 24 | 24 |
| Input | 20 | 0 |
| Unconditional | 14 | 0 |
| User | 10 | 0 |
| Successful intermediate event | 5 | 0 |
| Other | 5 | 0 |
| All gates | 186 | 106 |

The final failure class divides into 41 entries labelled hard and 41 labelled soft. The eight gates whose names match review all fall in self-reported completion and are optional. The blind second-reader check agrees on 182 of 186 original gate classifications, with reported chance-corrected agreement 0.87. These checks support the distinction among trigger types without turning the controlled applied-edit gate into a claim about a universal production default.

## J.2 A FUNCTIONING CRITIC AT THE FINISH CLAIM

The latest OpenHands SDK 1.40.1 experiment tests two functioning critic configurations on 59 tasks with two repetitions each. Every cell emits 118 distinct scores over 118 score records, so this evaluation includes a quality-varying score channel. Table 18 reports both all-task resolution and conditional repair/damage counts. The configurations are intervention settings within one harness, not independent harnesses.

Table 18: **Functioning finish-critic comparison.** Review-enabled policies resolve three or four more tasks per cell. The conditional counts use only triggered tasks and therefore need not equal the all-task net. Repetitions reuse the same 59 tasks.

| Setting | Rep. | Resolved tasks | | | Triggered tasks | |
|---|---|---|---|---|---|---|
| | | Review | No review | Net | Repairs / damage | Tokens / firing |
| cfgA | 1 | 14/59 | 11/59 | +3 | 4/2 | 540 |
| cfgA | 2 | 14/59 | 10/59 | +4 | 3/2 | 546 |
| cfgB | 1 | 14/59 | 11/59 | +3 | 4/2 | 558 |
| cfgB | 2 | 13/59 | 10/59 | +3 | 3/1 | 564 |

The all-task differences favor review in each cell, from $3/59 = 5.1$ to $4/59 = 6.8$ percentage points. This is a positive counterexample to a claim that review cannot help. It does not estimate the same intervention as the three-call harness: the trigger is a finish claim, the critic is optional, and the policies execute separately. The earlier constant-score integration is therefore a historical configuration boundary, not the result of this functioning-critic study. The present result is consistent with positive evidence for agentic review (Wang et al., 2026a).

**What transfers.** The numerical tax and review effect depend on the trigger, candidate distribution, prompt, available evidence and budget. The transferable measurement principle is to count what an event commits, evaluate the corresponding continuation from a shared state where possible, and compare it with the available alternatives at a stated cost. Neither a source census nor a positive finish-critic result establishes that automatic review is optimal in all harnesses.

# K Limitations and scope

## K.1 Populations and interventions

The primary tax measurement uses a controlled, oracle-localized three-call harness, while the shared-patch review, sequential controller and functioning-critic studies use distinct populations and protocols. Context ceilings of 32k and 64k are not pooled. Supplying the files touched by the reference patch removes localization from the primary comparison without revealing replacement text. This setting isolates review but is not a complete agent leaderboard evaluation. Canonical mappings identify the benchmark instances used, but do not establish contamination-free evaluation.

## K.2 Attribution and statistical interpretation

Independent review and no-review reruns can diverge before review. On the repaired review-fired set, 19 of 20 discordant pairs have different pre-review patch bytes. Shared-patch forks remove that source of ambiguity, and the original prompt has a positive measured effect, but the harmonized prompt's interval includes zero. Repository tests remain an imperfect semantic oracle. Zero disagreements in the repeated identical-patch evaluations apply to those checked patches and do not guarantee deterministic evaluation everywhere.

Repeated task executions do not create new independent tasks. The source controller's 308 assignments include 305 canonical issues, while the disclosure study has 84 distinct canonical issue identities in the current crosswalk. We state weighting and denominators with each analysis. Supplied intervals and independently reconstructed bootstrap intervals are distinguished in the appendices. Pointwise intervals, exploratory follow-ups and parameter sweeps do not supply simultaneous guarantees or independently confirmed selection of a best setting. The interpretation does not depend on a claim of independently verified prospective registration.

## K.3 Controller verification and interpretation

The stochastic controller replays from its supplied 24-dimensional features, frozen linear heads, action order and random draws. The feature contract includes eight frozen representation channels whose encoder is outside the replay surface. Consequently, the verification covers the score and selection computation on recorded representations, not independent reconstruction of those representations from repository state. Predicting terminal branch success also differs from predicting incremental gain over retaining the same patch. The calibration diagnostics do not establish the latter. A direct calibration check would compare each action with stopping from the same state, under a common continuation, on tasks excluded from fitting. The predicted-risk filter is not a population harm guarantee.

The isolated disclosure comparison holds acquisition fixed and controls access to the observation. In contrast, the additional masking, always-review and cost-sweep continuations generally start from a later saved checkpoint than the primary learned and schedule runs. Their task-paired results are informative operating points, but they do not isolate a policy change at an identical prefix. The additional source repetition also leaves learned control versus the correctness gate and test-all unresolved. The distinction matters because task pairing alone does not hold the starting patch fixed.

### K.4 COST BOUNDARIES

Review-stage tokens measure expenditure within the review arm and are not automatically the savings from deleting that stage. Input tokens, cached-price accounts, realized deployment allocations, hidden evaluation costs and allocated hardware capacity are different measures. The source policy's follow-up summary charges differ from its joined episode allocations, while the new continuation charges reconstruct at their own boundary. We therefore retain both source accounts and do not infer a verified saving over always review or a cost-dominating frontier. Evaluation costs are disclosed separately from the policy deployment bill (Appendix G).

### K.5 TRANSFER BEYOND THE CONTROLLED HARNESS

The production census counts source-code gates rather than runtime invocations. Its reconciled classes and second-reader checks describe the inspected inventory, not the prevalence of any trigger in deployed agents. The functioning finish-critic experiment provides positive review-enabled comparisons in a second harness, but remains limited to 59 tasks and two configurations. The additional controller cohort and worker checks broaden the empirical scope without proving universal transport. Appendix J distinguishes these evidence types.