FAILURE MIGRATION IN GUI AGENTS: OUTCOME CHURN AND THE TRADE-OFFS OF REFLECTIVE RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Higher aggregate success can conceal task-level regressions and successful states lost during retry. We study these two forms of outcome change and their resource trade-offs through paired OSWorld evaluations, same-configuration reruns, and reflective recovery. Scaling Qwen3.5 from 4B to 9B raises recorded full-split success from 21.4% to 26.3%. Among 342 tasks with valid evaluations for both models, 35 improve and 22 regress; an always-on accessibility scaffold yields 21 improvements and 31 regressions on 339 evaluable pairs. Same-configuration reruns also exhibit substantial outcome churn. We evaluate grounded verification and reflective retry with an always-strong agent and FMART, a self-hosted 4B agent with failure-triggered strong-tier takeover. The always-strong reflective system improves full-split success from 51.5% to 59.1% on 369 tasks; its gain remains significant in a runner-matched paired comparison. With a three-round cap, FMART reaches 57.2%; its retained records contain $0.77\times$ as many strong execution steps as the always-strong reflective system. Both record approximately \$0.42 per included task in their cost ledgers, which combine API estimates with nominal local-step charges for the cascade. In two FMART runs, five and seven tasks, respectively, have successful checkpoints followed by zero-score selected endpoints. Each of these tasks contains a successful checkpoint labeled incomplete by the completion checker. Together, these results show why evaluating GUI-agent improvement requires paired outcomes, returned-state success, and separate execution and monetary resource measures.

1 INTRODUCTION

GUI agents observe screenshots and execute mouse and keyboard actions in long-horizon environments such as OSWorld (Xie et al., 2024) and WindowsAgentArena (Bonatti et al., 2024). Perception scaffolds can enrich observations with visual marks or parsed interface elements (Yang et al., 2023; Lu et al., 2024). We examine two interventions: a larger vision–language model (VLM) backbone and an accessibility-tree scaffold. A higher aggregate success rate, however, does not say whether an updated agent preserves previously observed successes. That distinction matters for regression testing and for choosing where to spend additional inference compute.

We use *failure migration* as a descriptive name for task-level success/failure transitions between evaluated configurations. The central measurement is a paired outcome table: observed unlocks b, regressions c, and net change $(b-c)/n$. Long-horizon rollouts are stochastic: the same configuration can both unlock and regress tasks when rerun. We report paired intervention comparisons alongside same-configuration reruns and jointly examine tasks observed in both runs of each model.

On the 369-task OSWorld split, changing a Qwen3.5 backbone from 4B to 9B increases recorded success by $+4.9$ percentage points. On the 342 jointly evaluable tasks, this comparison produces 35 unlocks and 22 regressions. Adding an always-on accessibility-tree scaffold to the 4B agent produces 21 unlocks and 31 regressions on 339 evaluable pairs. Figure 1 displays unlocks, regressions, and net changes for these paired cohorts. The controlled scale comparison covers one model family; the frontier system is evaluated separately as a complete execution stack.

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.

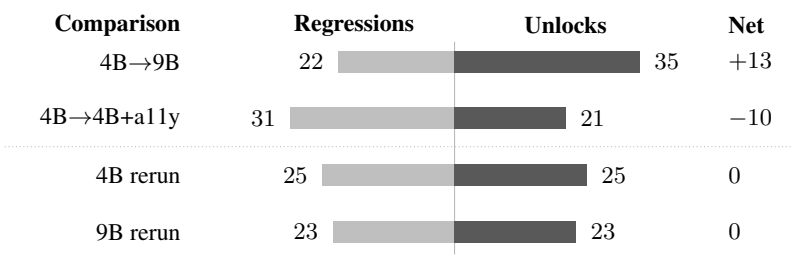

Figure 1: Paired outcomes with valid benchmark evaluations. Scaling and scaffold comparisons use 342 and 339 tasks; the 4B and 9B reruns use 339 and 355 tasks. Intervention comparisons (top) and same-configuration reruns (bottom) both contain unlocks and regressions. Net counts summarize the aggregate change, whereas total flips quantify changes in the observed success set.

The same preservation question arises within recovery: do later attempts retain tasks already completed by earlier attempts? A text-trajectory audit identifies premature completion and repeated ineffective actions as the predominant labeled symptoms. These observed symptoms motivate examining completion checking and retry. We evaluate an always-strong (all-CU) reflective system and FMART, a 4B-first cascade with strong-tier takeover. The former produces the clearest aggregate accuracy gain. Relative to all-CU reflect@3, FMART has fewer logged strong execution steps at similar selected-record ledger cost. The stopping audit traces successful checkpoints followed by failed selected endpoints in two FMART runs, connecting logged completion judgments to subsequent benchmark outcomes.

Contributions. We make three empirical contributions. First, paired intervention comparisons and same-configuration reruns quantify net improvement and success-set turnover (Section 4). Second, a checkpoint-to-endpoint audit across two FMART runs traces both recovered tasks and successful states lost during retry, linking these losses to logged completion judgments (Sections 5–6). Third, system comparisons at a common nominal round limit distinguish strong execution use from selected-record ledger cost (Section 6). Together, these measurements connect outcome changes across configurations, success preservation during recovery, and the resource trade-offs of hybrid execution.

2 Related Work

Outcome regression. Prediction churn and negative flips have been studied in classifier updates (Milani Fard et al., 2016; Yan et al., 2021), LLM updates (Echterhoff et al., 2024), and agent release evaluation (Zhang, 2026). Inverse scaling concerns a related aggregate question (McKenzie et al., 2023). Our study connects cross-configuration outcome changes with within-run success preservation during GUI recovery. Same-configuration reruns contextualize the former; checkpoint-to-endpoint audits measure the latter.

GUI agents, reflection, and routing. OSWorld, WindowsAgentArena, AndroidWorld, and WebArena provide interaction benchmarks (Xie et al., 2024; Bonatti et al., 2024; Rawles et al., 2024; Zhou et al., 2024). Work on GUI agents spans visual grounding (Wu et al., 2024; Lin et al., 2024; Yang et al., 2024) and multi-step execution (Xu et al., 2024; Qin et al., 2025; Agashe et al., 2025). Reflexion and Self-Refine use iterative feedback (Shinn et al., 2023; Madaan et al., 2023); external feedback can be distinguished from unaided self-correction (Huang et al., 2024; Gou et al., 2024). STEVE uses step-level verification to supervise computer-use agent training (Lu et al., 2025). VAGEN combines trajectory evidence with environment probing through a tool-augmented verifier (Cui et al., 2026). VeriGUI studies action-effect verification and self-correction, while VLAA-GUI combines stopping, recovery, and search mechanisms (Zhang et al., 2026; Han et al., 2026).

Adaptive VLM Routing proposes action-level model selection for computer use (Liu et al., 2026). Step-level optimization uses learned monitors to trigger strong-model escalation (Wei et al., 2026). These methods relate to cost-aware cascades, request routing, and model/tool orchestration (Chen et al., 2023; Ong et al., 2024; Su et al., 2025). FMART uses fixed coarse failure triggers and at

most one strong takeover per round. Our evaluation examines paired outcome changes, reflective stopping, and execution-resource use across the evaluated OSWorld systems.

Additional inference and selection. Repeated sampling can increase oracle coverage even without reflection (Brown et al., 2024). For computer use, Behavior Judge selects among rollout narratives and evaluates generalization across desktop and mobile environments (Gonzalez-Pumariega et al., 2025). This provides a non-oracle approach to selecting among multiple attempts. Our measurements examine the final-state consequences of sequential stopping and reset, including tasks completed in an earlier round but unsuccessful at the selected endpoint.

3 Evaluation Protocol and Measurements

Task sets and models. We evaluate the standard 369-task OSWorld split (commit `fe8c78e`). The controlled base-model comparison uses `Qwen/Qwen3.5-4B` and `Qwen/Qwen3.5-9B`, served with vLLM, with screenshots as input. The scaffold intervention adds the linearized OSWorld accessibility tree to the 4B planner. The strong executor and completion checker use Gemini 3.5 Flash. Its request builders leave the thinking option unspecified. Model and controller settings and historical-version coverage are listed in Appendix A.

Full-split success counts retain tasks without a selected success in the denominator. Nine tasks with persistent setup or evaluation failures are excluded from the reported 360-task headline analysis set. Record selection first follows the historical infrastructure filter and chronological order. Paired outcome analyses then require a completed numeric benchmark evaluation in both selected records. Evaluation exceptions remain missing, even when the logger stores their Boolean outcome as false. This validity filter holds the selected record fixed; it does not substitute an earlier attempt. The original all-CU arm has 359 selected records, of which 348 have valid evaluations. Content-policy refusals with completed evaluations retain their measured outcomes. Benchmark success includes correct termination on tasks configured as infeasible. Success-rate denominators, paired-test denominators, and cost denominators are stated separately (Appendix C).

Paired outcomes and reruns. Let $Y_{i,A}, Y_{i,B} \in \{0,1\}$ be the observed outcome for task i under configurations A and B. We report $b = \sum_i \mathbf{1}[Y_{i,A} = 0, Y_{i,B} = 1]$, $c = \sum_i \mathbf{1}[Y_{i,A} = 1, Y_{i,B} = 0]$, and

$$\widehat{\Delta}_{A\to B} = (b-c)/n, \qquad \widehat{C}_{A,B} = (b+c)/n. \tag{1}$$

The first estimates the aggregate success difference under the evaluated protocol; the second describes the observed outcome disagreement. We use exact two-sided McNemar tests on discordant pairs and 95% paired-task bootstrap percentile intervals. We evaluate the empirical resampling distribution directly from the task-level n, b, c counts (Appendix D). These intervals condition on the recorded rollouts; they do not include fresh-run or provider-version uncertainty. A same-configuration pair has zero expected signed net under exchangeability but can have substantial disagreement. Reruns provide empirical context for interpreting intervention-associated churn.

Headline point estimates use the original run, labeled seed 0. Independent reruns, labeled seed 1, exist for 4B, 9B, all-CU pass@1, and FMART reflect@3. These labels identify runs. The all-CU rerun spans a runner revision. Four cross-seed pairings share runs and are reported descriptively. The strongest accuracy evidence comes from all-CU reflect@3 versus all-CU pass@1; the scaling and cascade-versus-baseline contrasts provide supporting context. Unless labeled otherwise, p values are nominal. We additionally report a post-hoc Holm sensitivity analysis (Holm, 1979) over the nine contrasts in Table 3.

Success, selection, and resource axes. Reflect@N is configured to return the state at its stop or round limit, without benchmark-based round selection. The analysis selects the last admitted clean record and separately measures any-success coverage over retained records. FMART has continuous clean ranks from round 1 through its selected endpoint for 337/360 tasks; 23 histories have gaps. Its shorter prefixes therefore count retained clean records, not always actual controller rounds. Cost is the selected cumulative ledger: list-price API estimates plus a nominal local-step charge. Measured GPU expenditure is outside this ledger. Strong steps count logged `gemini_cu` execution steps, excluding auxiliary calls and execution absent from the archive. Appendix C gives the selectors, populations, and accounting terms.

4 Observed Outcome Churn and Failure Symptoms

Scaling and the scaffold intervention. The 4B→9B comparison increases recorded full-split success from 21.4% to 26.3%. On 342 evaluable pairs, $b = 35$, $c = 22$: net $+13/342$ ($+3.8$ pp), McNemar $p = .111$, bootstrap CI $[-0.6, +8.2]$ pp, Holm-adjusted $p = .526$. The cross-run scaling comparisons are reported on their jointly evaluable cohorts in Appendix D.

The scaffold comparison changes recorded full-split success from 21.4% to 19.8%. Its 339 evaluable pairs contain 21 unlocks and 31 regressions (net -2.9 pp, $p = .212$, CI $[-7.1, +1.2]$ pp). We find no detectable net improvement for this implementation. The intervention changes both input content and its length together and was evaluated in one run.

Same-configuration outcome churn. The 4B and 9B reruns flip $50/339$ and $46/355$ evaluable paired outcomes. The all-CU near-matched repeat flips $74/345$ outcomes, and the FMART reflect@3 repeat flips $70/353$. Each has zero signed net change ($p = 1$) (Appendix D). Stable aggregate success therefore coexists with substantial task-level disagreement. Paired reporting distinguishes net improvement from changes in the observed success set.

Outcome patterns repeated across runs. On the 337 tasks with valid evaluations in both runs of both base models, eight fail in both 4B runs and succeed in both 9B runs; four show the reverse pattern. Another 81 tasks change outcome across runs of at least one model. Table 9 gives the joint counts. The twelve tasks with repeated opposite outcomes provide concrete targets for regression testing alongside the run-variable tasks.

Text-trajectory failure audit. Claude Opus 4.8 labels the original selected non-success pools of the bare models (268 for 4B; 262 for 9B), using the recorded subgoal, action, and state-change trace. Gemini 3.5 Flash supplies a cross-check on the $213/196$ cases that pass its safety filter. The original base-model screenshots were overwritten by the benchmark's shared-file recording scheme; this audit is therefore text-trajectory-based, without an independent human gold standard.

The primary judge labels premature completion plus thrashing in $67.9\%/70.6\%$ of these traces; the cross-check gives $66.2\%/72.9\%$ on its labeled subsets. These historical pools include $5/2$ records with evaluation exceptions. The primary judge's capability-category shares are $6.3\%/4.2\%$, with conditional Wilson intervals $[4.0, 9.9]\%/[2.4, 7.4]\%$. Assigning every safety-blocked cross-check case to this category raises its shares to $20.5\%/26.0\%$. Full composition and agreement measures are in Appendix E.

Both audited non-success pools have similar symptom composition, dominated by premature completion and thrashing. These marginal distributions characterize separate non-success pools; thrashing itself can reflect insufficient planning or grounding capability. The labels therefore describe observable symptoms. Separately, the cheap rule-based Failure-Mode Identifier (FMI) has low agreement with the strong judges ($\kappa \approx .02$–$.05$). The evaluated system uses coarse failure triggers rather than FMI's fine-grained category routing.

5 Evaluated Reflective Systems

FMART: start locally, then hand over execution. The 4B base plans and executes actions until completion, termination, or a coarse failure trigger (CU-Pilot). Repeated ineffective actions, a stall threshold, or a rejected base-model completion claim can trigger the strong computer-use agent. A handoff diagnosis selects continuation from the current state or a clean reset; the strong model then owns the remainder of that round. The controller permits at most one takeover per round. The repository also contains category-specific FMI routing and proactive tool dispatch, but a historical inspected subset contains no invocations of those paths (123 files, 493 task records, approximately 14,900 steps).

Grounded stopping and reflective retry. A reflective run can reset and retry the entire cascade for up to three rounds. When another round and budget remain, the between-round checker uses the current final screenshot, accessibility state, and action trace. An incomplete verdict provides a short note for the next round. Because resetting can discard a completed state, FMART's stop controller

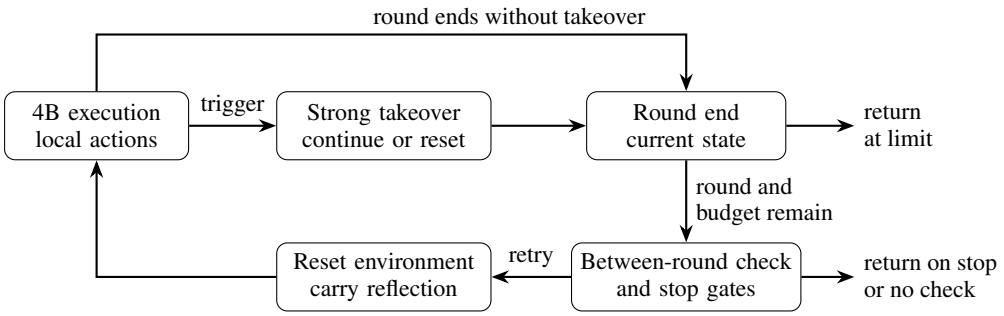

Figure 2: FMART execution and continuation. After takeover the strong model owns the rest of the round. Continuation depends on checks and safeguards, not the benchmark grader. Limit exits, unavailable checks, and some stop gates can return an unverified state (Appendix B). All-CU uses a different reflective controller.

requests a second judgment for a rejected or unusable done claim and can stop after two consecutive done-claimed rounds are rejected. These are heuristics for trading false accepts against destructive retries.

At the round or budget limit, the current state is returned without requiring a fresh positive between-round verdict. Missing screenshots or verification exceptions can also terminate continuation. Two unusable judgments on a done claim can result in trusting the claim. The benchmark evaluates correctness independently of these controller decisions. Appendix B specifies these branches and distinguishes within-round done checking from the between-round stop rule. The configured FMART budget is shared across rounds; accounting and affordability checks use estimated costs.

Always-strong comparator. All-CU executes from the start with the strong model. Its reflective variant uses completion checking and failure-analysis notes between attempts, with no oracle round selection. In the available implementation, its `run_reflective` callback omits FMART's trace-balanced second-opinion and consensus gates. We compare these systems at the same nominal round limit. The all-CU contrast measures the observed gain of the reflective system over its single-attempt reference; the FMART contrast compares both execution allocation and controller design.

Table 1: Controller comparison from the available-source audit. Appendix B gives the detailed control flow and Appendix A documents historical-version coverage.

Decision	FMART reflect@3	All-CU reflect@3
Execution	4B first; at most one strong takeover per round	Strong model from the start
Recovery	Retry the cascade with reflection notes	Retry strong execution with failure-analysis notes
Completion logic	Trace-conditioned checking; second opinion; consecutive-done consensus	Completion callback; no FMART-specific second-opinion or consensus gates
Budget handling	Estimate-based task budget shared across rounds	Inspected call site does not forward FMART's shared budget object
Returned result	State at the actual stop or round limit	State at the actual stop or round limit

Execution allowance and retained information. The available FMART configuration sets a nominal 50-step base round. A continuation handoff uses the remaining allowance, whereas a reset handoff gives the strong executor a fresh allowance, increasing the available action opportunities. Across reflective rounds, the controller carries text notes into a reset environment; optional prior-

image seeding is off by default. Table 1 summarizes the comparison, and Appendix A lists the available configuration and its provenance limits.

The returned state is the evaluation target. Let $Y_{i,r}$ denote benchmark completion after round r and $\tau_i \leq N$ the round at which the controller stops. The deployed and oracle outcomes over a complete execution history are different quantities:

$$S_i^{\text{return}} = Y_{i,\tau_i}, \qquad S_i^{\text{oracle}} = \max_{1 \leq r \leq \tau_i} Y_{i,r}. \tag{2}$$

The first evaluates the state actually returned to the user. The second credits any completed round, even if a subsequent reset loses that state, yielding diagnostic oracle coverage. Archive-based estimates use the last admitted clean endpoint and any-success coverage over available records. Missing histories and stop logs prevent establishing that this selected endpoint is the actual terminal state for every task (Appendix C). Reflection carries information across resets; preserving success also requires retaining the completed environment state.

6 System Results and Resource Trade-offs

Table 2 separates selected endpoints from retrospective clean-record prefixes. FMART@1/@2 select the last of the first one/two available clean records, without inspecting benchmark success. Because 23 histories omit earlier or intermediate ranks, prefix length counts available records. The second-record prefix has the same total wins as the selected final records at lower ledger cost (Appendix C).

Table 2: Observed operating points. Full-split success uses 369 tasks; the second rate uses the reported 360-task headline analysis set. Ledger cost averages one selected cumulative total per included task within each arm and includes nominal local-step charges. Step ratios are relative to all-CU reflect@3 and exclude auxiliary model calls; only the reflect@3 rows have matched nominal round limits. A dash indicates that this quantity is not reported. FMART@1/@2 are available-clean-record prefixes, including any checking already charged to those records; 23 task histories have gaps.

System	Success /369	Success /360	Ledger $/task	Logged strong steps
Base-model configurations				
4B bare	21.4%	–	–	–
9B bare	26.3%	–	–	–
4B+a11y	19.8%	–	–	–
Selected endpoints of single-attempt and three-round systems				
all-CU pass@1	51.5%	52.8%	0.24	–
FMART@3	57.2%	58.6%	0.42	0.77×
all-CU reflect@3	59.1%	60.6%	0.42	1.00×
Available-clean-record prefixes of FMART				
FMART@1 (prefix)	54.2%	55.6%	0.28	–
FMART@2 (prefix)	57.2%	58.6%	0.37	0.67×

Accuracy gains with the always-strong system. All-CU reflect@3 solves $218/369$ tasks versus $190/369$ for pass@1 (59.1% versus 51.5%). On their 347 jointly evaluable tasks, the paired difference is $+51/-26$, or $+7.2$ pp ($p = .006$; bootstrap CI $[+2.3, +12.1]$ pp). The full-split difference is $+7.6$ pp; the paired estimate uses only tasks with valid evaluations in both selected records. The original baseline predates a transient-API-retry correction. Against the post-correction rerun on 352 evaluable pairs, the reflective system solves 215 versus 191 ($+46/-22$, $+6.8$ pp, $p = .005$). The accuracy gain therefore persists against the runner-matched single-attempt reference. The two contrasts share the same all-CU reflective run and test robustness to the baseline revision. Across the nine-contrast Holm sensitivity analysis, their adjusted p values are .047 and .044, respectively; the remaining contrasts have adjusted $p \geq .083$ (Table 3).

Cascade operating points. The FMART second-clean-record prefix solves $211/369$ tasks (57.2%), a full-split gain of $+5.7$ pp over all-CU pass@1. Its paired 348-task evaluable contrast is $+51/-35$ ($+4.6$ pp, $p = .105$; CI $[-0.6, +9.8]$ pp; Holm-adjusted $p = .526$). Within FMART,

the final selected records differ from the first available records by $+16/-4$ on 355 evaluable pairs ($p = .012$). The second and final recorded prefixes have equal success counts but different solved-task sets (Appendix D). The three-round cascade and all-CU reflect@3 differ by 37 all-CU-only and 30 FMART-only wins ($p = .46$). The all-CU-minus-FMART difference is $+2.0$ pp, with a 95% paired-bootstrap interval of $[-2.5, +6.5]$ pp on 355 evaluable pairs (Appendix D). The interval spans meaningful advantages for either system.

Table 3: Paired contrasts with valid evaluations, oriented $A \to B$: b is B-only wins and c is A-only wins, as in Section 3. CU1/CU3 denote all-CU pass@1/reflect@3; FM1/FM2 are FMART clean-record prefixes and FM3 is its selected final endpoint. CU1r is the runner-matched baseline rerun. n is the jointly evaluable denominator; $\Delta = 100(b-c)/n$ is the net change in percentage points. p_{H} is the post-hoc Holm adjustment over these nine rows.

$A \to B$	n	b	c	Δ (pp)	Nominal p	p_{H}
4B → 9B	342	35	22	+3.8	.111	.526
4B → 4B+a11y	339	21	31	−2.9	.212	.635
CU1 → CU3	347	51	26	+7.2	.006	.047
CU1r → CU3	352	46	22	+6.8	.005	.044
CU1 → FM1	347	46	39	+2.0	.515	.928
CU1 → FM2	348	51	35	+4.6	.105	.526
CU1 → FM3	346	50	32	+5.2	.060	.359
FM1 → FM3	355	16	4	+3.4	.012	.083
FM3 → CU3	355	37	30	+2.0	.464	.928

Accuracy and success-set turnover. Table 3 connects the reflective results to the outcome-churn analysis. For all-CU, the 25-task paired net gain comprises 77 changed outcomes on 347 evaluable tasks: 51 gains and 26 losses. The second-record and final FMART results likewise have equal full-split success counts but different solved-task sets (Appendix D). Their $51/35$ and $50/32$ discordant pairs in Table 3 come from separate comparisons against all-CU pass@1, each on its own jointly evaluable cohort.

Execution allocation and ledger cost. The FMART second-clean-record prefix has a recorded cost of \$0.37 per included task, $1.54\times$ the single-attempt reference; selected reflect@3 cost is \$0.42, $1.75\times$. The retained FMART records contain 8,809 strong execution steps versus 11,503 for all-CU over the shared 360-task set: approximately 24.5 versus 32.0 per task, or $0.77\times$. Of FMART's 14,875 retained execution steps, 6,066 (40.8%) lack the strong-executor marker and carry the nominal base-step charge. No strong execution step is recorded for $61/360$ tasks (16.9%); these tasks may still call an external verifier. The $0.67\times$ prefix ratio also uses all-CU reflect@3 as its reference. All step counts cover the retained records.

The cost ledger combines list-price, uncached API estimates with the controller's nominal \$0.001 local-step charge. It includes charged auxiliary calls, retains charges already carried into a selected cumulative total, and does not separately add excluded recovery records or measured GPU expenditure. Execution-step counts measure where actions are generated; ledger cost also includes charged auxiliary computation.

Cost accumulation and the averaging population. Let $C_{i,r}$ be the cumulative ledger cost stored in record r and r_i^* the selected record index. For the arm-specific set of included tasks with a selected record $\mathcal{C}_A$, we report

$$C_i^{\text{selected}} = C_{i,r_i^*}, \qquad \overline{C}_A = \frac{\sum_{i \in \mathcal{C}_A} C_i^{\text{selected}}}{|\mathcal{C}_A|}. \tag{3}$$

The cumulative field is read once at the selected record; adding it across rounds would count earlier usage repeatedly. The averaging set is arm-specific: \$0.42 describes a mean over included tasks, while the execution-step ratio describes the allocation of logged execution. Appendix C specifies the record filters and excluded recovery costs.

Preserving benchmark success during recovery. Recovery adds successes but can also lose previously completed states. On 355 tasks with evaluable first and final FMART records, $195/199 = 98.0\%$ of first-record successes are retained, and $16/156 = 10.3\%$ of first-record failures are recovered. The original archive contains 217 tasks with a successful checkpoint and 211 successful selected endpoints. Five end with valid zero scores; one ends with an evaluation exception after an earlier zero score. The independent rerun contains 216 tasks with a successful checkpoint and 209 successful selected endpoints. All seven differences are valid zero-score endpoints. On its 354 evaluable first/final pairs, recovery adds 20 successes and loses 7 (Appendix E.3).

Round-level verdict errors. Of 478 retained FMART records, 428 contain a between-round decision note. Excluding four error/trust fallbacks leaves 424 usable notes; three lack valid evaluation scores. Table 4 reports the 421 usable, evaluated decisions. Logged-complete precision is $202/286 = 70.6\%$ and logged-incomplete precision is $125/135 = 92.6\%$. Ten of 212 successful checkpoints are labeled incomplete (4.7%). These are composite round-level decisions that can include a second opinion. The 50 records without notes are reported separately.

Table 4: Logged FMART between-round decisions versus benchmark success for 421 records with usable notes and valid benchmark evaluations. Four fallback notes, 50 missing notes, and three evaluation exceptions are excluded. The unit is a recorded round-end decision. This is a post-development audit.

Verdict	Benchmark success	Benchmark failure	Total
Complete	202	84	286
Incomplete	10	125	135
Total	212	209	421

Task-ID linkage connects these record-level errors to the endpoint audit. Seven of the ten incomplete labels on completed checkpoints belong to the six histories with an archived success but no selected success; two precede a later successful selected endpoint, and one labels a successful selected endpoint itself. Every observed success-loss history contains a completed checkpoint labeled incomplete before a later unsuccessful record (Table 12). In the independent rerun, all seven tasks with successful checkpoints and zero-score endpoints also contain incomplete labels at successful checkpoints. This association recurs across both runs.

One concrete case is a benchmark-designated infeasible task asking for audio translation from a still image. Its first round issues a `FAIL` action and receives score 1. The checker labels that response incomplete, with its logged reason focusing on missing audio extraction. The two later rounds contain no `FAIL` action and receive score 0. Completion checking must recognize valid termination as well as successful artifact creation.

7 Discussion and Limitations

Implications for evaluating GUI-agent improvements. The archived results motivate three complementary checks. Report unlocks and regressions alongside net success, with same-configuration reruns to characterize stochastic variation. For sequential recovery, evaluate the state actually returned and separately measure completed tasks lost along the way. For hybrid execution, report strong execution steps and ledger cost with their own inclusion rules and denominators. The $35/22$ evaluable scaling contrast, success losses in both FMART runs, and $0.77\times$ logged strong-step ratio illustrate why all three are needed.

Implications for recovery design. A completion checker recognizes successful states; a recovery controller decides whether to continue execution. The observed success-loss histories highlight their interaction. Checkpoint-level label accuracy alone does not capture whether later execution preserves or loses a completed state. Auditing state transitions and selected endpoints therefore complements evaluating the checker's judgments.

Scope and next experiments. The controlled interventions cover one model family, one scaffold, and OSWorld. Two runs per base model reveal repeated outcome patterns; estimating persistent task-level regression probabilities requires more repetitions. The failure audit uses model-assigned text-trace labels; screenshot-preserved human judgments would test their validity.

The reflective comparisons measure complete systems with different controllers and inference opportunities. A same-checker, budget-matched independent-retry control would isolate the contribution of reflection notes. A fixed-controller tier swap and stop-gate ablations would separate execution allocation from stopping logic. Prospectively run two-round policies would complement the retrospective record prefixes. These controls remain unperformed.

The archive supports per-task outcomes and ledger accounting. Missing round records, worker logs, and runtime model/version manifests limit reconstruction of complete execution histories. In particular, the loss histories link logged decisions to subsequent recorded outcomes; causal effects of changing a stop gate require a policy comparison. The numerical supplement exposes the observed task sets and sequences for independent analysis (Appendix F).

8 Conclusion

In the evaluated OSWorld systems, higher success rates coexist with task regressions, and same-configuration reruns also show substantial churn. The always-strong reflective system's accuracy gain remains statistically significant against a runner-matched baseline and under the stated multiplicity sensitivity analysis. Relative to all-CU reflect@3, FMART has fewer logged strong execution steps at similar selected-record ledger cost. Across two FMART runs, five and seven tasks have successful checkpoints followed by zero-score selected endpoints. Each history contains a successful checkpoint labeled incomplete. These findings support reporting paired outcomes, success preservation, and resource denominators alongside aggregate success.

AI Use Statement

Generative AI tools assisted with literature discovery, manuscript drafting and editing, formatting, feedback on analysis methodology and interpretation, statistical calculations, trajectory and tabular-data processing and auditing, and reproduction-code development. Experimental agents, verifiers, and model-based failure annotators are described in the paper. The author takes responsibility for the final text, claims, and artifacts.

Reproducibility Statement

Appendices A–B document the available model settings and execution controllers. Appendix C specifies record selection and resource denominators, and Appendix F lists the analysis scripts and required artifacts. The source package includes per-task and per-record numerical ledgers and a checker for paired counts, exact and Holm-adjusted p values, bootstrap intervals, recorded success coverage, and usable decision-note counts. The ledgers were regenerated from a checksum-verified archive of 405 JSONL files; one malformed line was explicitly skipped. The archive does not contain worker logs, all round histories, or historical runtime model/version manifests. Raw text traces and private repository identifiers are not included in the anonymous source package.

Ethics Statement

The experiments operate in OSWorld benchmark virtual machines. Tasks requiring unavailable external credentials fail at setup. Provider SAFETY blocks are handled using a fixed statement of the authorized sandbox context; remaining content-policy refusals retain their observed benchmark outcomes. Reflection can repeat actions and resets can discard completed work. These mechanisms should not be transferred to live systems with irreversible side effects without additional authorization, recovery, and safety controls. Hybrid execution can send observations to external models, including on tasks without strong execution takeover; it should not be interpreted as a data-residency guarantee.

REFERENCES

Agashe, S., Wong, K., Tu, V., Yang, J., Li, A., and Wang, X. E. Agent S2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

Bonatti, R., Zhao, D., Bonacci, F., et al. Windows Agent Arena: Evaluating multi-modal OS agents at scale. *arXiv preprint arXiv:2409.08264*, 2024.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Chen, L., Zaharia, M., and Zou, J. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Cui, C., Huang, J., Wang, S., Zheng, L., Kong, Q., and Zeng, Z. Agentic reward modeling: Verifying GUI agent via progressive trajectory-grounded interaction. *arXiv preprint arXiv:2602.00575*, 2026.

Echterhoff, J. M., Faghri, F., Vemulapalli, R., Hu, T.-Y., Li, C.-L., Tuzel, O., and Pouransari, H. MUSCLE: A model update strategy for compatible LLM evolution. *Findings of EMNLP*, 2024.

Gonzalez-Pumariega, G., Tu, V., Lee, C.-L., Yang, J., Li, A., and Wang, X. E. Scaling agents for computer use. *arXiv preprint arXiv:2510.02250*, 2025.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. CRITIC: Large language models can self-correct with tool-interactive critiquing. *ICLR*, 2024.

Han, Q., Tu, H., Wang, Z., Dai, H., Zhou, Y., Lau, N., Cardenas, A. A., Xu, Y., Xu, R., Xiong, C., Zheng, Z., Yao, H., Zhou, Y., and Xie, C. VLAA-GUI: Knowing when to stop, recover, and search, a modular framework for GUI automation. *arXiv preprint arXiv:2604.21375*, 2026.

Holm, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *ICLR*, 2024.

Lin, K. Q., Li, L., Gao, D., et al. ShowUI: One vision–language–action model for GUI visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

Liu, X., He, B., Liu, X., Luo, A., Zhang, H., and Chen, H. Adaptive Vision-Language Model routing for Computer Use agents. *arXiv preprint arXiv:2603.12823*, 2026.

Lu, F., Zhong, Z., Wei, Z., Liu, S., Fu, C.-W., and Jia, J. STEVE: A step verification pipeline for computer-use agent training. *arXiv preprint arXiv:2503.12532*, 2025.

Lu, Y., Yang, J., Shen, Y., and Awadallah, A. OmniParser for pure vision based GUI agent. *arXiv preprint arXiv:2408.00203*, 2024.

Madaan, A., Tandon, N., Gupta, P., et al. Self-Refine: Iterative refinement with self-feedback. *NeurIPS*, 2023.

McKenzie, I. R., Lyzhov, A., Pieler, M., Parrish, A., Mueller, A., Prabhu, A., McLean, E., et al. Inverse scaling: When bigger isn't better. *TMLR*, 2023.

Milani Fard, M., Cormier, Q., Canini, K., and Gupta, M. Launch and iterate: Reducing prediction churn. *NeurIPS*, 2016.

Ong, I., Almahairi, A., Wu, V., et al. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Qin, Y., Ye, Y., Fang, J., et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Rawles, C., Clinckemaillie, S., Chang, Y., et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*, 2023.

Su, H., Diao, S., et al. ToolOrchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*, 2025.

Wei, J., Ni, K., Zhao, Y., Gan, G., and Cohan, A. Step-level optimization for efficient Computer-use agents. *arXiv preprint arXiv:2604.27151*, 2026.

Wu, Z., Wu, Z., Xu, F., et al. OS-Atlas: A foundation action model for generalist GUI agents. *arXiv preprint arXiv:2410.23218*, 2024.

Xie, T., Zhang, D., Chen, J., et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *NeurIPS*, 2024.

Xu, Y., Wang, Z., Wang, J., et al. Aguvis: Unified pure vision agents for autonomous GUI interaction. *arXiv preprint arXiv:2412.04454*, 2024.

Yan, S., Xiong, Y., Kundu, K., Yang, S., Deng, S., Wang, M., Xia, W., and Soatto, S. Positive-congruent training: Towards regression-free model updates. *CVPR*, 2021.

Yang, J., Zhang, H., Li, F., et al. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023.

Yang, Y., Wang, Y., Li, D., Luo, Z., Chen, B., Huang, C., and Li, J. Aria-UI: Visual grounding for GUI instructions. *arXiv preprint arXiv:2412.16256*, 2024.

Zhang, D. AgentDevel: Reframing self-evolving LLM agents as release engineering. *arXiv preprint arXiv:2601.04620*, 2026.

Zhang, Y., Xue, X., Wu, X., Chen, M., Liu, C., He, X., Shao, R., Liu, F., Xu, H., Pan, Q., and Wang, H. Don't act blindly: Robust GUI automation via action-effect verification and self-correction. *ACL*, 2026.

Zhou, S., Xu, F. F., Zhu, H., et al. WebArena: A realistic web environment for building autonomous agents. *ICLR*, 2024.

A Configuration and Development Scope

Table 5 documents the available `configs/osworld_gem_fmart_reflect3.yaml` and active execution path, separately identifying unused routing machinery. The recovered records preserve task settings and sample ranks. Historical model IDs, source commits, and provider revisions require the unavailable versioned launch and request manifests. The base-model comparison uses the reported 32,768-token serving context. Base-model thinking is enabled in the FMART configuration. Gemini's execution and critic request builders leave the thinking option unspecified. Historical request manifests are needed to identify the effective provider-side setting; the available token-accounting helper includes returned thought tokens in output usage.

Table 5: Available FMART reflect@3 configuration and active controller settings.

Item	Value or behavior
Base model / serving	Qwen3.5-4B / vLLM; 32,768-token context
Base generation	Temperature 0.6, top-p 0.95, top-k 20; up to 16,384 new tokens; history images 0
Environment	OSWorld, AWS, PyAutoGUI; 1920×1080; accessibility tree available to checking/takeover
Nominal execution limit	50 steps per base round; takeover may continue with the remaining allowance or reset with a fresh allowance
Reflective rounds	Up to 3; text reflection notes; optional prior-image seeding is off by default
FMART task budget	$3.00 shared across rounds; estimate-based checks
Coarse takeover triggers	Three no-progress indications; stall at $\max(8, \lfloor 0.5T_{\max} \rfloor)$; rejected base completion claim
Handoff	Strong-model diagnosis chooses continue/reset; fallback resets after step 12
Between-round safeguards	Trace-conditioned judgment; second opinion on rejected/unusable done claims; two-done consensus stop
Unused attribution path	Category-specific FMI routing and proactive tool dispatch; no invocations in the inspected run records

The stop rule was developed on OSWorld pilot tasks, making the confusion matrix a post-development audit on the same benchmark.

B Execution and Stop Controllers

B.1 FMART: implemented control flow

The following pseudocode summarizes FMART's execution path in the available source. Benchmark evaluation is reserved for analysis; the completion checker supplies feedback to the controller.

Algorithm 1: FMART with at most N reflective rounds

Initialize one task budget, empty reflection notes, and done-rejection streak $s = 0$.
For round $r = 1, \ldots, N$:
  Reset the task and run the base with earlier reflection notes.
  On an eligible coarse trigger, diagnose continue versus reset;
  hand control to the strong executor for the rest of this round.
  Save the round result and cumulative cost ledger.
  If $r = N$, no strong agent exists, or no budget remains: return.
  Obtain the current final screenshot and available accessibility state.
  If no screenshot exists or verification raises: return.
  Judge completion with the action trace and balanced prompt.
  For a rejected or unusable done claim, apply the second-opinion gate
  in Table 6; charge the auxiliary calls.
  If the resulting verdict is accepted (including trusted-claim fallback): return.
  If the round ended in a done claim, increment s; otherwise set $s = 0$.
  If $s \geq 2$: return the current state (consensus stop).
  Append the incomplete verdict as a note for the next reset round.
Return the current state.

Table 6: FMART second-opinion gate for a done-claimed round. "Unusable" means the verifier returns its error/parse-failure acceptance sentinel. Consensus and limit exits are additional branches outside this table.

First judgment	Second judgment	Gate result
Accept	Not requested	Stop
Reject or unusable	Valid accept	Stop
Reject or unusable	Valid reject	Incomplete; may retry
Reject	Unusable	Retain first reject
Unusable	Unusable	Trust done claim; stop

The second opinion calls the same model again. For a round without a done claim, an unusable acceptance sentinel permits continuation. Within-round checking is distinct: a rejected base done claim can cause takeover in the same round. Between-round checking determines whether to reset the whole cascade. A takeover reset gives the strong executor a fresh step allowance. Effective action opportunities therefore depend on handoff mode.

B.2 ALL-CU: SEPARATE REFLECTIVE CONTROLLER

In the available implementation, all-CU uses the strong agent's `run_reflective` method; FMART uses `run_task_reflective` in the meta-agent. The former runs a strong-model attempt, checks a done claim through a completion callback, and, if continuation is required and another attempt remains, obtains a failure-analysis note and resets. It concatenates the action logs and returns the final state. It does not use FMART's trace-balanced second-opinion or consecutive-done consensus gates. Its call site also does not forward FMART's shared task-budget object to the executor. Table 1 summarizes these controller and budget differences.

C RECORD SELECTION, COST, AND DENOMINATORS

Record selection. The analysis identifies the task universe from the 4B arm and removes nine predefined task-ID prefixes associated with persistent setup or evaluation failures. A clean record has no recognized infrastructure error in its top-level `error` field and contains steps or an observed final-success field. Records are ordered by `ended_at`, then sample rank. The selected endpoint is the final clean record in the admitted sequence. Selection uses record order, infrastructure-error status, and the presence of an observed result. The infrastructure-error filter includes reset, step/plan failure, and evaluation failure/timeout/error. The predefined exclusions and error filter are shared across the headline analyses. Evaluation-signal validity is checked separately after selection.

The phenomenon comparison uses a separate legacy selector: prefer a record without any error, then the latest occurrence in sorted-file and physical-row order. It retains a task whose only record has an error; an absent task remains missing. The headline and clean-rerun analyses instead apply the infrastructure filter and use actual clean intersections. The numerical supplement retains both conventions and their task IDs and denominators.

The recovered archive contains 405 checksum-verified JSONL files and $4{,}121$ parseable records. One malformed line is excluded explicitly, without repairing it or interpreting it as a failed task. Among FMART's 360 retained seed-0 sequences, 337 have consecutive sample ranks starting at zero; 23 have missing earlier or intermediate ranks. The retained record counts are $280/42/38$ for lengths one/two/three, whereas the last retained ranks are $0/1/2$ for $263/47/50$ tasks. These are different measurements: record length is not physical stop round.

For the standalone all-CU reference, the admitted sequence can include SAFETY and infrastructure recovery runs; the last clean standalone record is selected. Its "pass@1" label describes the selected single-attempt execution policy. The collection campaign also includes retry, mopup, and escalation attempts. Both systems receive the fixed authorized-sandbox request context. The reported recovery changes the all-CU result from 31.7% to 51.5%: 73 of the 123 initially SAFETY-blocked tasks are recovered; remaining refusals stay failed.

Cumulative cost is read once. Equation 3 defines the selected cumulative cost and its arm-specific averaging set $\mathcal{C}_A$. Summing $C_{i,1} + \cdots + C_{i,r_i^*}$ would count earlier rounds repeatedly. The recovered per-round records confirm shared-budget accumulation; `parity_report.py` reproduces the selected-record accounting. For example, one three-record task has cumulative costs \$0.764283, \$1.096941, and \$1.9799985. Its first-step costs are \$0.001, \$0.765283, and \$1.097941: each later first-step value carries the previous cumulative total plus the nominal local-step charge. Across all 112 adjacent retained record pairs with consecutive ranks, that increment is \$0.001. Auxiliary calls can be charged directly to the shared budget, so summing visible step costs is not in general equal to the record's total. All-CU's concatenated reflective execution similarly reports the whole admitted reflective run, whereas the single-attempt reference selects one standalone recovery record.

Costs include metered execution, diagnosis, verification, reflection, and within-episode SAFETY rephrasing when charged to the selected record. Earlier standalone recovery and infrastructure-failed records are not added separately. Any charges already carried into a selected cumulative total remain included. Accounting for the full recovery campaign requires identifying additional excluded-attempt charges without double counting. Standalone restart overhead and within-run cumulative round costs are different quantities; the latter must not be re-summed to estimate the former.

Within the released parseable standalone records, all-CU pass@1 costs \$88.703889 in total versus \$86.525529 selected: the difference is \$2.178360 (2.52% of selected cost). For all-CU reflect@3, the corresponding totals are \$151.240347 and \$151.224582, a difference of \$0.015765. These compare archived records, not invoices or a complete campaign manifest. FMART's per-round fields sum to \$227.6021235, but this includes repeated cumulative usage and cannot estimate restart overhead. Missing ranks and the malformed line prevent a complete FMART restart-cost reconciliation from this archive.

What a recorded prefix measures. Let $m_i \leq 3$ be the number of retained clean FMART records for task i, with outcomes $Z_{i,j}$ and cumulative costs $L_{i,j}$ in retained order. A prefix of length K uses $j_i(K) = \min(K, m_i)$ and reports

$$S_i^{\text{prefix}}(K) = Z_{i,j_i(K)}, \qquad C_i^{\text{prefix}}(K) = L_{i,j_i(K)}. \tag{4}$$

The success field is read for evaluation, not for choosing $j_i(K)$. Thus this is not the oracle coverage $\max_{j \leq j_i(K)} Z_{i,j}$. Neither the prefix success nor the returned final-state success must increase with K: later execution can replace a successful earlier state. This is why equal aggregate wins at $K = 2$ and $K = 3$ do not imply identical solved tasks. For the 23 gapped histories, j does not equal physical round rank. Consequently these prefixes cannot establish the success rate of a two-round cap across all 360 tasks.

The stored cumulative cost can include the selected record's between-round checker and second opinion, because the original run had another round available. In the available controller, setting the cap to K would exit before these calls. We retain the historical ledger values rather than subtracting an unmeasured per-task checking cost. The \$0.28 and \$0.37 entries are therefore recorded-prefix costs, not separately measured invoices for newly configured one- or two-round policies.

Denominators. The full-split success rate divides wins by 369; the headline analysis-set rate divides by 360. A paired test requires the stated common task set. In contrast, mean cost divides by the number of clean included records for that arm. For example, the historical all-CU selected total of \$86.53 gives \$0.2410 over 359 included records, but \$0.2345 when spread over all 369 tasks. These are different normalizations, not two estimates of different token usage. The reported \$0.24/\$0.37/\$0.42 means and $1.54\times$ /$1.75\times$ ratios use the record normalization. The original all-CU arm has 359 selected records, including 11 without numeric benchmark scores. Outcome comparisons use the jointly evaluable sets in Table 3; resource accounting retains costs from selected records regardless of evaluation availability.

Cost-ledger components. The tracker prices API input tokens without modeling provider caching and also assigns a nominal \$0.001 charge per local execution step. The 6,066 visible non-strong steps alone contribute \$6.066 in nominal charges; earlier missing steps may also be represented in cumulative totals. The API component uses list-price uncached estimates, whereas the local component is a proxy rather than measured GPU expenditure. A historical Vertex spot check found actual billing around 0.42 times the metered amount, consistent with cached input being priced

at 10% of uncached input. That spot check is not a complete, per-arm billing reconciliation, its invoices are absent from this archive, and its factor cannot be applied to the entire mixed ledger. More repeated strong-model context may favor all-CU under caching, but no measured cached-billing ratio between the two full arms is established here. Self-hosted base compute (historically reported as approximately 185 GPU-hours) is not measured by the nominal step charge. Hosting, wall-clock latency, concurrency, and energy were not compared.

Strong-executor step counts. The step numerator counts execution records whose `notes` start with `gemini_cu`. It does not count completion-verifier, handoff-diagnosis, or reflection calls as additional execution steps. Those calls can still contribute cost. The three-round ratio is $8{,}809/11{,}503 = 0.766$ over the reported shared 360 tasks. The second-clean-record prefix ratio 0.67 uses the same three-round all-CU reference. Neither number implies a corresponding reduction in total requests, input tokens, latency, or externally exposed task data.

As a missing-history sensitivity check, the 337 tasks with consecutive retained FMART ranks have 8,205 strong steps versus 10,774 for all-CU reflect@3 (0.762×). Their mean ledger costs are \$0.4133 versus \$0.4210 and their selected success counts are 201 versus 202 (30 FMART-only, 31 all-CU-only). This post-hoc, telemetry-defined subset preserves the direction of the logged-step comparison; consecutive retained ranks do not prove complete campaign histories or establish a prospective cost advantage.

D Paired Results and Rerun Context

Table 3 in the main text reports the nine paired contrasts. Here we give rerun context and the details of the multiplicity check.

Table 7: Independent reruns. Gross disagreement is substantial while the observed signed net is small. The all-CU pair spans a runner revision; it is not an exact same-version repeat. A high p value does not prove equivalence. Each row uses its jointly evaluable task set. Here A is seed 0 and B is seed 1, fixing the direction of b, c.

System	Flips	b	c	Net	p
4B ($n = 339$)	50	25	25	0	1.0
9B ($n = 355$)	46	23	23	0	1.0
all-CU pass@1 ($n = 345$)	74	37	37	0	1.0
FMART reflect@3 ($n = 353$)	70	35	35	0	1.0

Multiplicity sensitivity from published counts. For each row of Table 3, the exact nominal value is $\min\{1, 2\Pr[\text{Binomial}(b+c, 1/2) \le \min(b, c)]\}$. We apply Holm's step-down adjustment (Holm, 1979) to the unrounded values for the existing nine-row table. If $p_{(1)} \le \cdots \le p_{(9)}$, the adjusted value at rank k is $\min\{1, \max_{j \le k}(10 - j)p_{(j)}\}$. The two all-CU reflective contrasts remain below .05; scaling, the FMART-versus-baseline contrasts, and the within-FMART later-round contrast do not. Holm adjustment does not require independent contrasts, but valid input tests are still required. The family was specified retrospectively from the reported table and does not cover every cross-seed comparison, pilot, or choice of policy. This is a bounded sensitivity analysis, not prospective family-wise error control for the whole research process. The script packaged with the manuscript reproduces these calculations from the printed counts. Task-bootstrap intervals in the main text are unadjusted and do not model within-application task dependence or fresh-run variability.

A paired-bootstrap interval from aggregate counts. For binary task outcomes, the paired difference $D_i = Y_{i,B} - Y_{i,A}$ takes values $-1, 0, 1$. Its empirical resampling distribution is fully specified by n, b, c:

$$\overset{*}{\Pr}(D = -1) = \frac{c}{n}, \quad \overset{*}{\Pr}(D = 0) = 1 - \frac{b + c}{n}, \quad \overset{*}{\Pr}(D = 1) = \frac{b}{n}. \tag{5}$$

The bundled checker computes the distribution of $\Delta^* = n^{-1} \sum_{j=1}^{n} D_j^*$ by repeated convolution and takes its 2.5th and 97.5th percentiles. For FMART@3→all-CU reflect@3, $(n, b, c) =$

$(355, 37, 30)$ gives $[-2.5, +6.5]$ pp after rounding. This evaluates the empirical bootstrap distribution without Monte Carlo sampling; it is not an exact finite-sample coverage guarantee. All paired success-difference intervals use task-level counts and this same procedure. For example, all-CU pass@1→reflect@3 has $(n, b, c) = (347, 51, 26)$ and interval $[+2.3, +12.1]$ pp. The per-task checker verifies the input counts against the numerical ledgers. Resampling conditions on these rollouts and treats task pairs as units; it does not model run-level or application-level dependence.

Cross-seed comparisons and cohort reconciliation. Table 8 applies the phenomenon selector, then intersects tasks with valid evaluations in both arms. The paired task sets differ across comparisons. The numerical supplement preserves the original stored flags and identifies evaluation availability separately.

Table 8: Exploratory 4B→9B cross-seed comparisons under the phenomenon selector and valid evaluations. The same runs recur across pairings; these are not four independent replications.

4B seed → 9B seed	n	4B wins	9B wins	b	c	p
0 → 0	342	79	92	35	22	.111
0 → 1	340	79	93	38	24	.098
1 → 0	355	83	97	30	16	.054
1 → 1	354	83	98	33	18	.049

The FMART-versus-baseline cross-seed comparisons use 345–351 evaluable pairs and have nets of 17–19 ($p = .060/.089/.041/.048$). The same runs recur across pairings, so this is a stability check, not four independent confirmations. With several exploratory comparisons, the near-threshold p values warrant particular caution.

Joint outcomes across two runs per base model. Table 9 intersects the four evaluable task sets before inspecting outcomes. Each task contributes once, indexed by its number of successes in the two 4B runs and in the two 9B runs. This keeps all observations for a task together instead of treating the four cross-run pairings as independent replications. On this common cohort, the 4B runs solve $79/79$ tasks and the 9B runs $92/92$. The within-model disagreements are 50 and 44 tasks, with 13 tasks in both sets, so $50 + 44 - 13 = 81$ vary for at least one model.

Table 9: Success counts across two recorded runs per base model on 337 jointly evaluable tasks. Each entry counts tasks. The off-diagonal corner cells identify opposite outcomes repeated across both runs.

	9B successes in two runs			
4B successes in two runs	0	1	2	Total
0	200	25	8	233
1	19	13	18	50
2	4	6	44	54
Total	223	44	70	337

The seed-0 comparison on this cohort contains 35 gains and 22 losses. Eight of those gains and four of those losses have the same direction in the seed-1 comparison. The other discordant tasks include run-to-run changes in one or both models. These counts condition on valid evaluations in all four runs and describe their joint outcome patterns.

Prefix outcomes and oracle coverage. FMART's 478 retained records give an average of 1.33 records per task. The first retained record succeeds on 200 tasks and the selected endpoint on 211, under the stored Boolean convention. The second-clean-record prefix also has 211 successes, but not on the same task set. Across all available checkpoints, 217 tasks succeed at least once.

Third-record and actual third-round turnover. Under the original stored-flag convention, matching the 38 three-record histories gives two gains, two losses, and 34 unchanged non-successes between their second and third records. The physical-round comparison is different: 42 tasks have

both raw sample ranks 1 and 2, with two gains, three losses, and 37 unchanged non-successes. These counts include an evaluation exception among final non-successes. Four of these tasks lack a rank-zero record, so their physical third round is the second retained clean record.

Independent-run oracle comparison. An oracle union of the two independent all-CU pass@1 reruns covers $226/344 = 65.7\%$ tasks versus $213/344 = 61.9\%$ for the reflective system on their jointly evaluable subset (37 oracle-only versus 24 reflective-only, $p = .124$). This union measures available coverage across runs using benchmark outcomes to select attempts. A deployable independent-retry system would instead require its own completion-based selection rule.

E Failure-Symptom Audits and Stopping Decisions

E.1 Base-model failure labels

Table 10: Primary-judge label composition over selected non-success base trajectories (268 for 4B; 262 for 9B). Each column is conditional on that model's audit pool; differences are not paired transitions. Labels describe the judge's interpretation of text traces, not validated root causes or measured repairability.

Assigned symptom/category	4B (%)	9B (%)	Difference (pp)
Premature done claim	26.5	30.5	+4.0
Thrash / loop	41.4	40.1	−1.3
Grounding	10.4	13.7	+3.3
Wrong strategy	12.3	8.8	−3.5
Capability category	6.3	4.2	−2.1
Other / ambiguous	3.0	2.7	−0.3
Premature done + thrash	67.9	70.6	+2.7

Claude Opus 4.8 supplies complete primary-judge coverage; Gemini 3.5 Flash labels $213/196$ trajectories after its safety filter. On these subsets, the latter's capability-category shares are $0.0\%/1.0\%$ and behavioral shares $66.2\%/72.9\%$. Assigning every blocked case to the capability category would instead yield $55/268 = 20.5\%$ and $68/262 = 26.0\%$, quantifying sensitivity to the safety-filtered cases.

Six-way inter-judge agreement is $\kappa = .39/.57$. For the coarse remedy-mapping split, raw agreement is $92.5\%/94.4\%$ but $\kappa = .18/.24$, reflecting strong imbalance. Both judges use the same incomplete traces and may share errors. The heuristic remedy mapping associates $90.7\%/93.1\%$ of cases with proposed interventions; its practical recovery rate and cost remain unmeasured because these routes were not exercised.

The rule-based FMI agrees with the primary judge on $44.8\%/43.0\%$ of cases ($54.5\%/55.6\%$ against the cross-check), with $\kappa \approx .02$–$.05$. These values measure category agreement. The evaluated controller uses separate coarse no-progress triggers. Agreement denominators require jointly labeled subset IDs, which are absent from the available audit summaries.

The full split contains 290 tasks without recorded success for 4B and 272 for 9B. The primary audit covers 268 and 262 selected non-success traces, respectively, including 5 and 2 evaluation exceptions. Thus 22 and 10 full-split non-successes lie outside those audit pools. The text-only labels and different conditioning sets prevent reading the column differences as a task-matched transition in underlying capabilities.

For the scaffold intervention, the outcome matrix is: 47 shared wins, 32 4B-only wins, 26 scaffold-only wins, and 264 shared failures. Per-step FMI verdicts were not logged for that arm, so no comparable category-transition table is inferred.

E.2 Strong-tier audit and failure overlap

A same-family Gemini audit of 164 all-CU failures identifies wrong strategy and thrashing as the predominant symptoms. The recovered reference ledger contains 169 selected non-success records

Table 11: Original rule-FMI transition counts for 4B→9B, $n = 369$. P: perception; R: reasoning; L: loop/no-progress; O: no verdict. These labels have weak agreement with the model audit and are retained as recorded telemetry, not as validated failure-cause transitions. The historical stored-flag margins are 40 unlocks and 22 regressions.

Source / destination	Win	P	R	L	O
Win (79)	57	1	2	16	3
P (6)	1	0	0	3	2
L (242)	27	3	1	190	21
O (42)	12	0	0	15	15

and 10 tasks without a clean record, together accounting for 179 full-split non-successes. The audit's subset IDs are absent from this release, so its coverage cannot be joined to that ledger. As with the base audit, the labels describe text-trace symptoms, not validated root causes or evidence that a reset or reflection would solve a labeled task.

Under the original stored-flag convention, FMART@3 has 149 non-success endpoints, including four evaluation exceptions. On the 359 common clean tasks, 115 are unsuccessful under both systems and 33 are reference-only wins; the remaining FMART failure has no selected reference record. These historical overlap counts precede evaluation-validity filtering; the evaluable contrast is in Table 3. Among those 33, 30 involve continued rather than reset takeover and 3 are logged as missed escalation. This co-occurrence is not a controlled attribution of loss to handoff: comparing continue/reset on the same inherited states was not performed. Against these regressions, FMART has 54 unique wins. The second-clean-record prefix has $56/35$ rather than $54/33$ discordant outcomes despite the same net gain.

E.3 Stop-rule confusion

Table 4 in the main text gives the verdict counts.

The 478 retained records contain 428 completion notes. Four are explicit fallbacks: two error-accept notes on unsuccessful records and two trusted-done-claim notes on successful records. Excluding them gives 424 usable decisions. Three lack valid benchmark evaluations, leaving 421 evaluated decisions: $202/84$ complete and $10/125$ incomplete labels on successful/unsuccessful checkpoints, respectively. These are composite round-end labels: 18 include a second-opinion acceptance, not an independent verifier measurement. Completion sensitivity is $202/212 = 95.3\%$ and specificity for incomplete checkpoints is $125/209 = 59.8\%$. The 50 records without notes all have raw sample rank two; they are not assigned a synthetic verdict.

Logged verdicts and observed continuation are not interchangeable. Twenty early retained endpoints have an incomplete note (1 successful, 19 unsuccessful). Conversely, the two error-accept notes are followed by another retained record. Worker logs are needed to distinguish consensus exits, unavailable observations, and other branches. Classifying all early endpoints as verifier accepts and all continued records as rejects would produce a stop/continue partition, not a verifier matrix.

Verdict errors versus final-state losses. The ten usable incomplete labels on completed checkpoints occur on nine distinct tasks. Seven such events belong to six tasks whose selected endpoints record no success; five are valid zero scores and one is an evaluation exception. One task contributes two events. Two events are followed by another retained record and ultimately a successful selected endpoint. The remaining event labels a successful selected endpoint, with no later retained clean record. This linkage distinguishes errors on individual checkpoints from losses of tasks.

Table 12 lists every task that succeeds at an available checkpoint but has no selected success. Five histories contain all three sample ranks; one omits the initial rank. Each observed successful checkpoint in these six histories has a usable incomplete label and a subsequent valid zero-score record. These sequences establish the temporal association in the records, not which gate caused continuation or whether a modified gate would preserve success in a new run.

Let F_1, F_2, F_3 denote the available-clean-record prefix success sets. Their first/final shared-success count is $200 - 5 = 195$, so $|F_1 \cup F_3| = 200 + 211 - 195 = 216$. The analogous available-record

Table 12: Six original-run histories with success absent at the selected endpoint. IDs are public OSWorld task-ID prefixes. Round numbers equal the stored zero-based sample rank plus one. S/F denotes scored success/failure; E denotes an evaluation exception. I denotes a usable logged incomplete decision; – denotes no decision note. The last entry is the selected clean endpoint. Missing round 1 in one history is not imputed.

Task ID prefix	Recorded rounds	Outcomes	Logged decisions
`12086550`	1, 2, 3	S, S, F	I, I, –
`58d3eeeb`	1, 2, 3	S, F, F	I, I, –
`aceb0368`	1, 2, 3	F, S, F	I, I, –
`bc2b57f3`	1, 2, 3	S, F, F	I, I, –
`e246f6d8`	2, 3	S, F	I, –
`efcf0d81`	1, 2, 3	S, F, E	I, I, –

coverage is $|F_1 \cup F_2 \cup F_3|$. Direct task-ID matching gives

$$|F_1 \cup F_2 \cup F_3| = 217, \qquad |(F_1 \cup F_2) \setminus F_3| = 6. \tag{6}$$

Thus observed any-success coverage is $217/360 = 60.3\%$, compared with $211/360 = 58.6\%$ selected-endpoint success. The six-task gap measures observed success coverage absent at the selected endpoint; it comprises five valid zero-score endpoints and one unavailable endpoint evaluation.

The independent FMART rerun contains 514 retained records across 358 tasks, with continuous ranks from zero in each history. Its 355 evaluable endpoints include 209 successes and 146 failures; three endpoint evaluations are unavailable. Among 354 evaluable first/final pairs, 189 of 196 first-record successes persist and 20 first-record failures are recovered. All seven success-loss histories contain an incomplete label on a successful checkpoint. The rerun's 459 usable, evaluated decisions give a complete-label row of 200 successes and 89 failures, and an incomplete-label row of 18 successes and 152 failures. Its 18 incomplete-on-success events concern 16 tasks. Seven events belong to the seven success-loss tasks, six precede later successful endpoints, and five occur at successful selected endpoints.

F Reproduction Artifacts

The source package contains the manuscript, ICLR formatting files, and three offline checkers with privacy-minimized ledgers regenerated from the checksum-verified archive. All three legacy exports match the repository CSVs byte for byte. Analysis files are in `manuscript_checks/`:

- `evaluation_validity_check.py` recomputes paired statistics, four-run outcomes, rerun comparisons, and both runs' success-preservation and decision-label audits. Missing scores remain distinct from stored false flags; record selection is fixed.
- `paired_summary_check.py` reproduces historical aggregate counts before evaluation-validity filtering. `per_task_check.py` checks the four unchanged original exports and their stored-Boolean summaries.
- `data/` contains the task-level headline and phenomenon tables, clean-record sequences, logged verdicts, and source and derived-file hashes. Its README documents fields and privacy filtering.

All checks run offline using the Python standard library. The primary check is:

```
python3 -B manuscript_checks/evaluation_validity_check.py
```

With the runtime repository and raw archive, regenerate and analyze exports with:

```
python scripts/export_per_task.py TRAJ_DIR OUT_DIR
PYTHONPATH=.  python scripts/parity_report.py TRAJ_DIR
```

Packaged costs retain the exports' four-decimal rounding; Appendix C reports totals from raw records. The checkers reproduce packaged tables; agent execution uses the runtime repository.