COUNTING ROUNDS IS NOT MEASURING PROGRESS IN DAY-SCALE AGENT EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Longer agent runs produce more interactions, but what do those interactions add? We argue that neither work-order counts nor instruction labels can stand in for realized progress, and support this claim with three linked analyses of 93 day-scale coding episodes. First, 391 runtime attempts reduce to 283 distinct work orders; GPU-unavailable tasks contribute 83.7% of later orders but only 31.1% of recorded input tokens, exposing a strong mismatch between order counts and resource exposure. Second, two isolated runs of the same judge model agree that only 24 of 61 full-text extension instructions require behavioral change. Third, removing one selected recorded patch from each of six games reproduces behavioral failures that the retained patches prevent under matched conditions. Four of these repairs arose under extensions without a mandatory behavioral change, and one under a fallback continuation. The experiments separate repeatable patch-local effects from instruction semantics: a request to verify can lead to a repair, while another order need not mark another increment. We derive collection and evaluation implications from this distinction and preserve the underlying boundaries in reusable action, handoff, and next-order records.

1 INTRODUCTION

An agent that runs for a day may implement a substantial project, repeatedly revisit an unchanged one, or spend much of its budget waiting for a prerequisite. Long-horizon benchmarks evaluate demanding engineering and research tasks (Wijk et al., 2025; Chan et al., 2024; Starace et al., 2025), but interpreting the execution between the initial request and the final artifact remains difficult. How much of the later activity is additional useful work? This question matters both for allocating inference budgets and for deciding which trajectories to retain. Counting interactions or supervisor-issued instructions is convenient; treating those counts as progress risks rewarding the collection process rather than the work it produces. Our claim is that *execution boundaries and instruction labels must be linked to artifact-level evidence before they can support claims about useful progress*.

Three findings make this distinction consequential. In 93 nominally day-long coding episodes, retries inflate 283 work orders to 391 runtime attempts, and GPU-unavailable tasks account for 159 of 190 later orders. These tasks contribute only 31.1% of the corpus's 2.81 billion recorded input tokens: counting later orders and counting resource exposure describe markedly different populations. Next, full-text review finds that 37 of 61 instructions labelled as extensions do not require changed behavior. Yet controlled patch ablations recover concrete repairs from four such extensions and from a fallback continuation. The two results fit together: an instruction specifies what must be done, whereas execution determines what actually changes. Verification-oriented work can discover and repair a defect even when its instruction does not mandate a behavioral addition.

The findings expose two concrete errors that proxy-based analysis would make in this corpus. Reading later-order activity as marginal progress would conflate changing task composition with within-task change. Retaining only extensions that mandate behavioral additions would exclude four replayed repairs. The contribution is to demonstrate these distinct failure modes in the same collection and link selected instructions to the recorded actions and executable evidence needed to distinguish them.

We make three contributions:

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.

- **An order-aware audit of day-scale execution.** We reconcile attempts, work orders, environmental eligibility, and recorded usage, showing how retries and blocked tasks distort accounts indexed by later orders (Section 3).
- **A linked analysis of requested and realized work.** Repeated same-model instruction review and six authenticated patch ablations connect instruction semantics to measured behavioral effects, with targeted failure checks and ordinary-behavior controls (Section 4).
- **Evidence-preserving implications for collection and reuse.** We identify which boundaries must survive trajectory export and how the findings change the interpretation of continuation, verification, and resource use (Sections 5–6).

The organizing distinction is between *requested work*, *artifact change*, and *new evidence*. A new requirement belongs to the first category; a repaired interaction belongs to the second; a newly executed check on an unchanged artifact belongs to the third. All can matter, but they answer different questions. Figure 1 shows how we link these observations without forcing them into a single progress score.

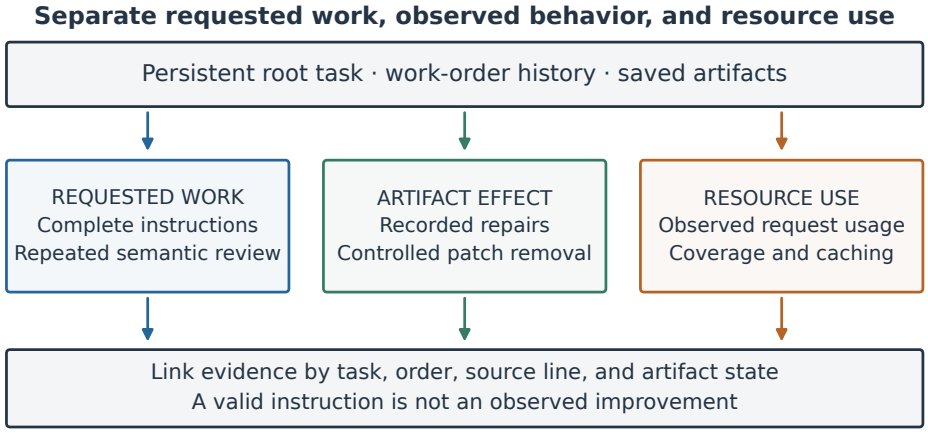

Figure 1: Linking requested work, artifact effects, and resource exposure. Full-text instruction review characterizes requirements; patch ablation tests a specific behavioral effect; usage records describe exposure. Task and order provenance connect the three analyses.

2 Units and Evidence of an Increment

An *episode* contains one root objective and a persistent workspace. A *work order* specifies an instruction and its acceptance checks; the initial order states the root objective, and a *later order* is any subsequent instruction. An *attempt* is a runtime session that may resume the same order after interruption. We reserve *continuation* and *extension* for the supervisor's actions: continuing unmet work and introducing further scope, respectively. A *fallback order* is a generic continuation emitted after supervisor-generation failure. Context resets change the visible conversation, potentially within the same attempt or order.

For work order o_r, let W_r^- and W_r^+ be the workspace states before and after execution. A behavioral increment is a difference in supported behavior between these states under specified conditions. It may satisfy a new requirement, repair an existing interaction, or improve reliability under a workload. An increment can occur entirely within an existing file. Conversely, a new file or a new order can leave the relevant behavior unchanged. Evidence also has a state: a passing check establishes something about the artifact on which it ran, and subsequent edits can make that observation stale.

This distinction determines the comparison needed. Instruction review asks what o_r requires; a boundary comparison asks how behavior differs between W_r^- and W_r^+. Our replay study instead compares the final artifact with and without one recorded patch, holding its other code fixed. It therefore measures a patch-local effect under final dependencies rather than reconstructing a historical

whole-workspace increment. The audit, review, and replay retain their respective units throughout the paper.

3 Later-Order Counts Reflect the Collection Process

3.1 Retries and fallback orders change what is being counted

The corpus comprises 58 GameCraft tasks, six pygame tasks, 11 CUDA language-model tasks, and 18 additional kernel tasks. GameCraft requests complete Godot 4 micro-games from empty projects. The 29 CUDA/kernel episodes lacked the required GPU; we distinguish them from the 64 hardware-eligible game episodes. A Kimi K3 teacher executed through Kimi Code CLI with nominal 24-hour budgets. The runtime imposed a separate 12-hour session fuse and re-issued unfinished orders, producing initial-order restarts in 67 episodes.

Merging attempts by persisted order identity yields **283 work orders from 391 attempts**: 93 initial and 190 later orders. Only 41 episodes receive a later order, while initial orders occupy a median of 23.6 hours, including waits and interrupted execution. Counting attempts as objectives would inflate the total by 38.2%. The difference is attributable to the execution mechanism: a fresh session can resume an unchanged instruction against the same workspace.

The 190 later orders also differ in how they arise. The supervisor generates 82 orders accepted by the runtime: 61 extensions after a recorded complete status, 17 continuations after incomplete status, and four after blocked status. The remaining **108 are fallback orders** after generation failure. Their default incomplete/continue fields encode the recovery path rather than a fresh assessment of project state. Thus even a count of unique later orders combines deliberate scope decisions with infrastructure-driven continuation.

3.2 High-index orders increasingly describe blocked tasks

GPU-unavailable episodes contribute **159/190 later orders** and 116 of 120 orders at index 3 or greater (Figure 2). The latter group has a median of zero newly logged paths. This concentration changes the interpretation of a declining activity curve: its right-hand side increasingly represents tasks whose required validation was unavailable. It is not repeated observation of the same representative task as additional work is supplied.

Eligible games show a different pattern. Their later orders contain a median of 45 assistant steps for GameCraft and 53 for pygame. Restricting to non-fallback extensions gives 10 GameCraft orders and 11 pygame orders, with medians of 59 and 60 steps. Substantial interaction therefore remains possible after the initial order. The contrast with blocked tasks shows why a pooled later-order count loses information: the same boundary can precede extended implementation, focused verification, or another attempt to overcome an unresolved prerequisite.

3.3 Resource exposure reverses the apparent emphasis

Across whole episodes, the GPU-unavailable group contributes **31.1% of recorded input tokens but 83.7% of later orders**. The token total is 2.81 billion input and 24.52 million output tokens, with usage available for 22,759 of 22,810 solver requests. An order-weighted account therefore emphasizes blocked work far more than a token-weighted account. These are different denominators: later-order counts concern transitions, whereas token usage covers entire episodes, including initial orders and repeated context. Appendix D gives exact totals and caching coverage. The contrast establishes the accounting mismatch without treating either token volume or order volume as utility.

4 Instruction Labels and Realized Repairs Answer Different Questions

4.1 Most extension instructions do not require behavioral change

We review all 61 supervisor-generated extension instructions with their full acceptance lists. The criterion is whether satisfying an instruction *requires new or changed artifact behavior*. A manda-

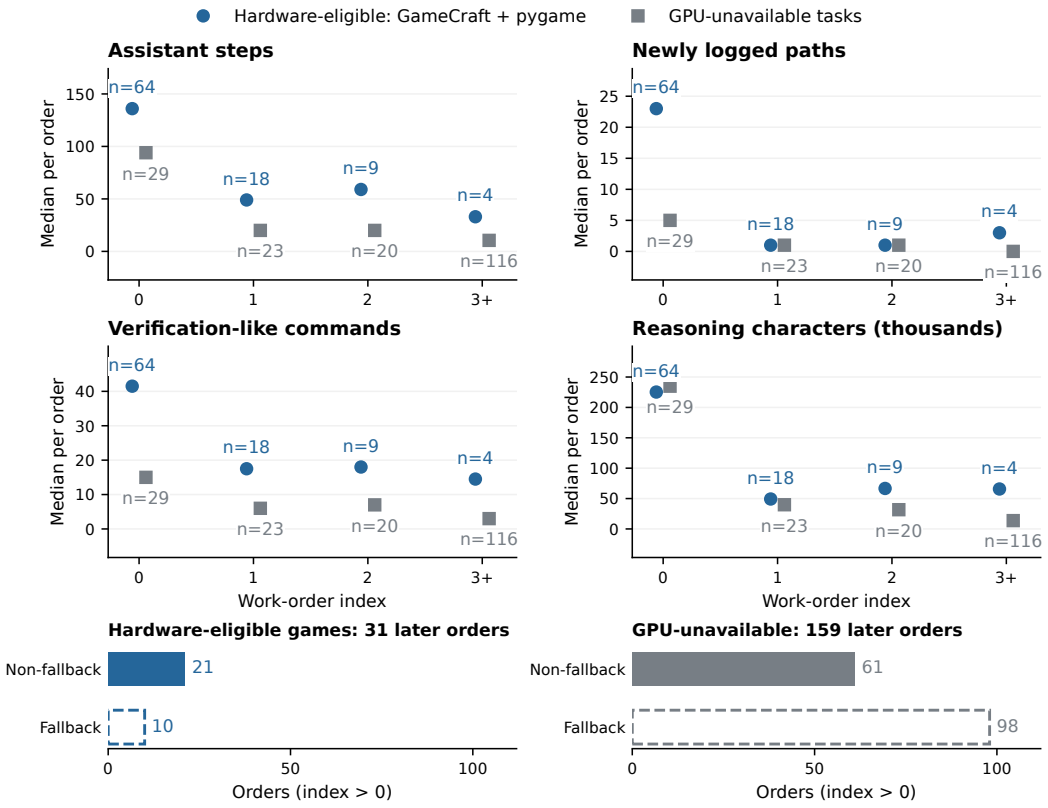

Figure 2: Activity by work-order index after merging retries. The number and composition of contributing orders change sharply: GPU-unavailable tasks supply 116 of 120 orders at index 3+. Points are subgroup medians; the bottom panels separate supervisor-generated and fallback orders. These unpaired activity summaries describe collection behavior, not a learning curve or independently judged artifact quality.

tory repair for a stated defect qualifies. A request to test and repair only if a defect is discovered does not: it can be satisfied by verification alone. This criterion separates required behavioral change from the recorded extension action.

Two isolated runs of the same judge model, `gpt-6-astra`, each review every full-text instruction at high reasoning effort. Neither receives prior labels, aggregate results, or the other run's output. Each returns yes/no/unclear, a short reason, and a verbatim evidence span; all identifiers and quoted spans are validated. Both runs assign **24 yes and 37 no labels**, with 61/61 agreement and Cohen's $\kappa = 1$. This establishes same-model repeatability, with calibration limits discussed in Section 8.

The split persists within eligible games: 12 of 21 extensions require behavioral change, leaving nine that request verification or conditional repair without a mandatory addition. The GPU-unavailable group contributes the other 12 positive labels among 40 extensions. An earlier clipped review assigned 23 positives; both full-text runs change only the compiler-configuration case, so excluding that case returns 23/61 without changing the finding. The broader initial review is retained in Appendix C.

The important consequence is not that the 37 negatively labelled orders were unproductive. Four selected repairs in the next section occurred under such orders. The negative label constrains what the instruction demands, not what the agent can discover while carrying it out. Using this label to discard all verification-oriented execution would remove observed instances of useful repair. Conversely, a positive label records an obligation whose fulfillment still requires checking the artifact.

4.2 AUTHENTICATED PATCH ABLATIONS MEASURE SPECIFIC BEHAVIORAL EFFECTS

Selection and experimental unit. We inspected all six pygame episodes and selected one replayable later-order repair from each. This domain provides Python game components, event loops, and archived checks that can be exercised in one headless replay environment. The GameCraft artifacts are Godot projects and require a separate engine-level replay setup; they remain in the corpus audit but are outside this experiment. Selection required saved final code, a reversible recorded edit, and a concrete diagnostic. Six patches were taken through authentication and paired replay; the exploratory inspection did not retain an exhaustive inventory of all candidate edits, so no candidate-level selection rate is available.

What the comparison measures. For each selected patch, we authenticate the complete recorded edit against its stored hash and require a unique match in the final source. We then reverse only that replacement, holding the remaining final code, inputs, and runtime fixed. Two edits require recovery of full arguments from the original traces because parsed previews are truncated. This procedure connects the tested change to an actual later-order action rather than to a newly invented repair. The pair answers: *under these conditions, what behavior depends on this recorded patch in the final artifact?*

Behavioral outcomes and controls. Each diagnostic targets the behavior affected by its patch. Seeds, input keys, frame times, or phase offsets vary the conditions within that diagnostic. We report which behavior fails, how many tested conditions expose the failure, and whether the repair prevents it. A separate ordinary-behavior control checks that the repaired version has not merely disabled the relevant subsystem. All controls pass in both variants: fresh input still moves ships, live-player encounters still schedule enemies, pickups still move and slow down, a nearby player still wakes the bat, separate-frame taps still move the character, and ordinary-step collision checks still succeed.

Table 1: Patch-local outcomes. "Removed" and "retained" count targeted violations. Probes are chosen for each patch, so the fractions describe reproducibility under tested conditions, not effect size or independent task successes. The final column states the observed behavioral change (removed → retained). V: extension without mandatory behavioral change; B: extension requiring it; F: fallback order.

Game	Order	Removed	Retained	Measured effect
Space Battle	V	20/20	0/20	Unintended movement after menu input → stationary until fresh input
Imps & Assassin	F	20/20	0/20	1–2 enemy-scheduling events after death → none
Dr E's Lab	V	20/20	0/20	Slowdown depends on update rate → consistent slowdown
Shadow Vale	B	20/20	0/20	Enemy stays asleep on ground approach → activates
Grid RPG	V	20/20	0/20	Short tap ignored → exactly one move
Green Ninja	V	3/28	0/28	2 missed projectile hits and 1 missed landing → none

Every selected patch prevents at least one reproducible violation (Table 1). Four patches come from extensions labelled V, one from a fallback order, and one from an extension labelled B. Figure 3 exposes the underlying measurements rather than pooling these different diagnostics into an overall success percentage. The results establish counterexamples to equating an order's requirement label with its realized behavioral contribution; they do not estimate repair prevalence or the benefit of a continuation policy.

4.3 WHAT CHANGES IN THE ARTIFACT, AND WHY THE CONTROLS MATTER

Input and state transitions. Space Battle's selected order asks for input checks and conditional repairs. The ensuing patch clears held and pressed keys at mode transitions. With the patch removed, holding a menu-navigation key through match start makes the ship move unintentionally in all 20

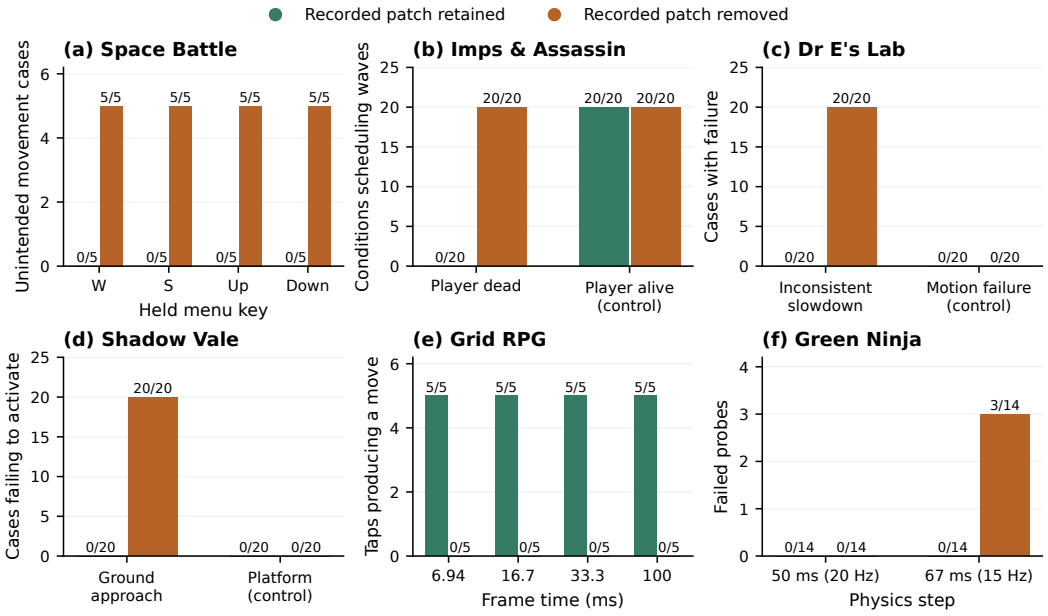

Figure 3: Direct behavioral outcomes with the recorded repair retained (green) or removed (orange). Panels report unintended movement, enemy scheduling after death, inconsistent slowdown, failure to activate an encounter, short taps producing a move, and collision failures. Fractions show affected cases within each tested condition; labelled controls check ordinary behavior. Other final code is fixed. Outcomes remain separate because the probes test different behaviors.

key/seed conditions. With the patch retained, the ship stays stationary until fresh gameplay input arrives (Figure 4). Separate controls start from rest and confirm that fresh input still works. The control distinguishes an input-state repair from globally disabling movement. The archived 86-test suite also passes on the retained artifact.

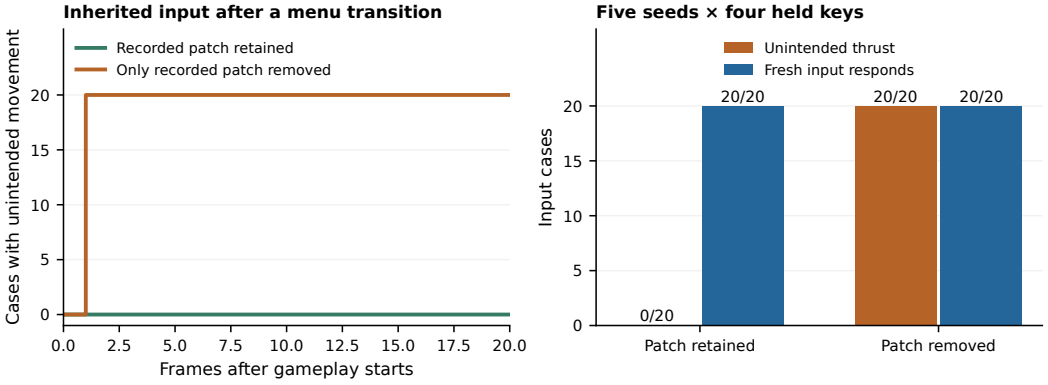

Figure 4: Space Battle: does menu input cause unintended gameplay movement? Left: the number of affected cases after gameplay starts, across five seeds and four held keys. Right: unintended movement and response to fresh input in separate controls starting from rest. The repair eliminates the unintended response while preserving normal input handling.

Grid RPG exposes a related but distinct event-ordering error. A movement press and release can arrive in the same frame. Removing the patch lets release handling erase the press before it produces a turn: all 20 taps are lost across frame times of 6.94, 16.7, 33.3, and 100 ms. With the patch retained, each tap produces one turn and one tile of movement. Separate-frame controls succeed in both versions. Thus the repair preserves short events rather than increasing movement indiscriminately.

Imps & Assassin tests state validity after death. Its selected patch, produced during a fallback order, guards enemy scheduling after game over. Across four spawner phases and five seeds, removal produces one or two new-wave or portal scheduling events during 120 updates after lethal damage; retention produces zero. Separate live-player controls schedule normally in all 20 conditions, and all 31 archived tests pass on the retained artifact. A fallback order lacks a new supervisor assessment, yet its execution can still contain a specific, testable repair.

Consistent interactions and reachable encounters. Dr E's Lab repairs a frame-rate-dependent slowdown of scattered pickups. Without the patch, the same amount of elapsed game time produces different slowdown depending on how often the game updates. The repair makes that behavior consistent across the tested update rates. All 20 diagnostic conditions expose the inconsistency without the patch and pass with it. Ordinary-motion controls pass in both versions, showing that the repair preserves moving objects rather than simply freezing them.

Shadow Vale's patch moves a sleeping bat down one tile, bringing a player on the ground route within its activation range. Without the patch, the bat stays asleep during every tested ground approach; with it, the encounter activates. Approaching on the nearby platform wakes the bat in both versions. This comparison identifies exactly what the layout repair changes: access to the encounter from the ground route, while preserving activation from the platform.

Green Ninja illustrates why the outcome fractions must be read with their stimuli. Removing physics substeps causes three failures among 28 archived large-step probes: two missed projectile hits and one missed ledge landing, all at a 67 ms update interval. The 50 ms probes and six ordinary 60 Hz projectile controls pass in both versions. Retaining substeps eliminates the three failures. Unlike the other selected diagnostics, this failure is exposed only at some tested phases and step sizes; the per-condition display preserves that difference.

Visible artifacts and behavioral evidence. Figure 5 shows the saved Space Battle artifact and its existing image checks. The project already has a rendered game scene and a passing 17-case image inspection before the selected input repair. The later replay asks a different question: what happens when a held key crosses the menu-to-match boundary? A static capture does not exercise that event sequence. Linking the request, edit, and replay therefore identifies a contribution that would be invisible in a comparison of screenshot counts or unchanged visual appearance. The prior capture establishes context; the matched execution establishes the patch's effect.

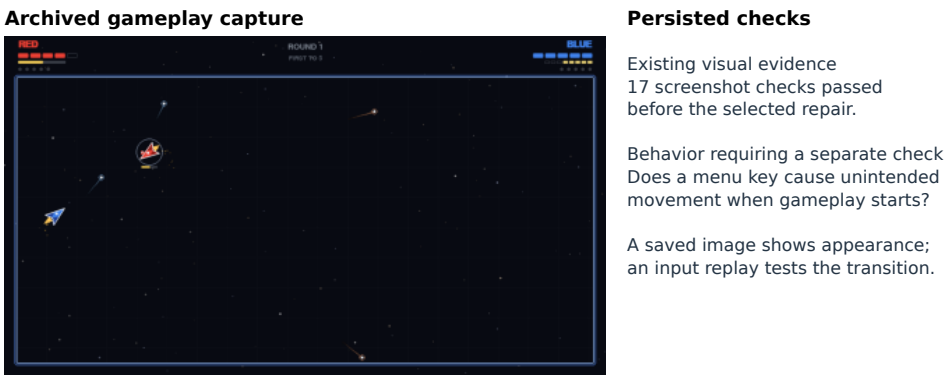

Figure 5: The artifact behind the input replay. The archived gameplay capture and image checks predate the selected repair. They provide visual context, not a before/after comparison; Figure 4 measures the later input-state effect.

5 Preserving the Evidence in Reusable Trajectories

The same provenance supports three reusable views. **Action records** preserve an order's preceding context and selected interactions, keeping tool calls with results and separating earlier context from target actions. **Handoff records** retain actual reset summaries, reconstructable progress updates, and

subsequent resumption. **Next-order records** pair the supervisor's visible evidence with its accepted non-fallback decision. All views retain root-task, order, conversation, and context-reset identities, so an instruction can be traced to subsequent actions without confusing a reset or retry with new scope.

The export contains 572 action samples, 30 real summary-writing events, 118 reconstructed progress targets, and 82 accepted next-order decisions. Real resets remain distinct from synthetic length-bounded windows; inserted state must precede the target actions, and overlapping views retain common task identities. These choices preserve what was assigned, what was visible, and what was executed. Construction details, handoff examples, and diagnostics are in Appendix A; prospective student assessment is in Appendix B. The export is an additional research artifact, not evidence of downstream capability gains.

6 Implications for Collection and Evaluation

An execution boundary is an observation about the collector. The 108 fallback orders and the concentration of high-index orders in GPU-unavailable tasks explain why a larger order count can arise without a larger set of completed objectives. Counting those events remains useful for diagnosing supervision and recovery, but their meaning depends on source and eligibility. The corresponding reporting unit is a work order linked to its predecessor, status evidence, and artifact state. This allows a collector's reliability to be assessed separately from the solver's contribution, rather than crediting repeated recovery as additional scope.

Verification is a route to progress, not a synonym for either progress or waste. The four V-labelled repairs show that instructions permitting verification alone can lead to changed behavior. The fallback repair extends the point: useful execution does not require a newly reasoned scope decision at every boundary. What makes these cases informative is the combination of a recorded change, an executable diagnostic, and preserved ordinary behavior. The six replays distinguish realized repair from the intention to seek one. Filtering trajectories by instruction labels alone would collapse this distinction and discard some of the very interactions that reveal latent defects.

The evidence unit should match the claim. The replay results have different meanings: a menu transition causes unwanted movement, a short tap is ignored, object motion depends on the update rate, or an encounter fails to activate on approach. Naming the failed behavior and checking that it is repaired makes the result interpretable. A single pooled pass rate would hide the difference between 20 variants of one trigger and three phase-sensitive collision failures. Likewise, 31.1% of whole-episode input tokens and 83.7% of later orders illuminate a denominator mismatch, not a cost-per-repair estimate. Artifact-local effects, task outcomes, and resource efficiency require corresponding comparisons.

Proxy-based selection can err in both directions. The corpus exposes two distinct selection problems. Preferring records with more later orders disproportionately selects GPU-unavailable episodes and supervisor recovery. Rejecting every extension that lacks a mandatory behavioral addition removes the four V-labelled repair cases. One proxy can overstate the evidence for progress, while another can exclude demonstrated local contributions. Neither problem is resolved by combining the two labels: order frequency describes how often control returned, and instruction semantics describes the obligation issued at that return. The missing observation is still the artifact effect. Keeping these signals separate allows instruction review to locate intended work and behavioral checks to establish what selected actions accomplished. This also avoids the opposite mistake of declaring every verification-oriented order valuable on the basis of a few replayable repairs.

These findings motivate a continuation specification that distinguishes completing unmet checks from adding new requirements. Existing checks remain valid objectives when evidence is missing; new scope should be explicit when required. The revised specification and prospective controlled comparisons appear in Appendix B. Its outcome advantage is untested; the present contribution is the evidence showing why these distinctions matter.

7 Related Work

Long-horizon evaluation and behavior analysis. RE-Bench, MLE-bench, PaperBench, and SWE-EVO assess demanding engineering outcomes (Wijk et al., 2025; Chan et al., 2024; Starace et al., 2025; Le et al., 2025). Our question concerns the interpretation of activity within extended execution: which observed boundaries correspond to a change in requirements, behavior, or evidence? Analyses of long-horizon failure and horizon length examine how extended tasks challenge agents (Wang et al., 2026; Kim et al., 2026). We add a collection-level source of ambiguity: the tasks represented at high order indices change with eligibility, return behavior, and supervisor failures. The patch study then links selected later actions to executable local effects, complementing aggregate measures of horizon or final success.

Task generation and trajectory quality. Recursive task synthesis, TaskCraft, AgentSynth, and synthetic coding environments construct tasks or environments for agent interaction (Li et al., 2026; Shi et al., 2026; Xie et al., 2026; Zhu et al., 2025). SWE-Gym, SWE-smith, and R2E-Gym develop executable software-engineering data and verification (Pan et al., 2025; Yang et al., 2025; Jain et al., 2025). Research on effective terminal trajectories asks which interaction data supports downstream agents (Yang et al., 2026). Our analysis addresses an earlier inference: whether an instruction or activity label is adequate evidence that useful work occurred. The V-labelled repairs show why semantic filtering and behavioral validation should remain distinct, even when the collected records are intended for later reuse.

Context continuity. Context-Folding, ReSum, MEM1, and Kimi K2.5 investigate mechanisms for sustaining work under limited context (Sun et al., 2025; Wu et al., 2025; Zhou et al., 2025; Kimi Team, 2026); Slipstream studies trajectory-grounded compaction validation (Chen et al., 2026). Our export preserves the distinction between actual resumption after a reset and a synthetic reduction of a longer history. This is an accounting and evidence requirement rather than a competing memory mechanism: continuity claims depend on what state the solver actually received and what it subsequently did.

8 Limitations

The corpus is small and game-heavy, and task domain is confounded with hardware eligibility. Order-index groups are unpaired and reflect the collection's return and failure mechanisms. Logged path and command counts are incomplete activity proxies. Same-model review agreement does not provide human calibration, and the initial broader review used clipped evidence. The six repairs were selected purposively after inspection, without an exhaustive candidate inventory; their diagnostic conditions are not independent tasks. Patch ablations hold final dependencies fixed and do not reconstruct historical whole-project gains, quantify overall game quality, or estimate population-level repair frequency. The Godot-based GameCraft domain has no corresponding replay experiment here.

No controlled comparison establishes that a revised supervisor, a longer budget, or adaptive scope improves outcomes. Provider-reported tokens include repeated context and caching, omit separate supervisor calls and 51 solver records, and do not measure active compute or monetary cost. Incompletely pinned service versions limit exact regeneration. Sparse real resets restrict continuity coverage; synthetic windows can omit dependencies, and overlapping exports are not independent replications. The separate 48-hour collection, documented only in the appendix, changed configuration and has not been reconstructed at order level. The present evidence supports a measurement distinction and selected local repairs, not an optimal collection or training strategy.

9 Conclusion

Counting later orders and classifying their instructions cannot replace checking what their execution contributes. In this corpus, retries and blocked environments distort order-based accounts, while verification-oriented and fallback execution contain reproducible repairs. Linking requested work to recorded changes, behavioral diagnostics, and resource exposure makes those contributions visible without confusing more activity with more progress.

AI USE STATEMENT

Generative AI produced the solver and supervisor trajectories and assisted with analysis, implementation, manuscript revision, and adversarial review. Instruction annotations used a judge model distinct from the trajectory generator; the full-text study repeated the same judge model in two isolated contexts. Patch outcomes and regression results come from executed checks. The authors remain responsible for verifying claims, citations, and submission requirements.

ETHICS STATEMENT

Raw trajectories and workspaces require access, licensing, and secret-removal review before release. Generated projects should be evaluated in isolated environments. An internally auditable artifact is not automatically redistributable, and we make no public-release commitment here.

REPRODUCIBILITY STATEMENT

The appendices specify collection units, extraction rules, review criteria, replay stimuli, controls, software versions, and usage coverage. Recorded edit hashes link the replayed changes to the collected trajectories. These support reproduction of the retrospective analyses; exact regeneration of the original teacher execution is limited by incompletely pinned service versions.

REFERENCES

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024. URL https://arxiv.org/abs/2410.07095.

Zhuofu Chen, Rui Pan, Yinwei Dai, and Ravi Netravali. Slipstream: Trajectory-grounded compaction validation for long-horizon agents. *arXiv preprint arXiv:2605.08580*, 2026. URL https://arxiv.org/abs/2605.08580.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv preprint arXiv:2504.07164*, 2025. URL https://arxiv.org/abs/2504.07164.

Sunghwan Kim, Junhee Cho, Beong-woo Kwak, Taeyoon Kwon, Liang Wang, Nan Yang, Xingxing Zhang, Furu Wei, and Jinyoung Yeo. On training large language models for long-horizon tasks: An empirical study of horizon length. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.02572.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. URL https://arxiv.org/abs/2602.02276.

Tue Le, Minh V. T. Thai, Dung Nguyen Manh, Huy Phan Nhat, and Nghi D. Q. Bui. SWE-EVO: Benchmarking coding agents in long-horizon software evolution scenarios. *arXiv preprint arXiv:2512.18470*, 2025. URL https://arxiv.org/abs/2512.18470.

Zhongzhi Li, Yucheng Shi, Zongxia Li, Ruhan Wang, Anhao Li, Zixun Huang, Junyao Yang, Lei Ke, Ninghao Liu, Haitao Mi, and Leowei Liang. Recursive synthesis for long-horizon terminal tasks. *arXiv preprint arXiv:2608.05466*, 2026. URL https://arxiv.org/abs/2608.05466.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.10055.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025. URL https://arxiv.org/abs/2510.11967.

Xinyu Jessica Wang, Haoyue Bai, Yiyou Sun, Haorui Wang, Shuibai Zhang, Wenjie Hu, Mya Schroder, Bilge Mutlu, Dawn Song, and Robert D. Nowak. The long-horizon task mirage? diagnosing where and why agentic systems break. *arXiv preprint arXiv:2604.11978*, 2026. URL https://arxiv.org/abs/2604.11978.

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

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025. URL https://arxiv.org/abs/2509.13313.

Jingxu Xie, Dylan Xu, Xuandong Zhao, and Dawn Song. AgentSynth: Scalable task generation for generalist computer-use agents. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.14205.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.21798.

Sidi Yang, Chaofan Tao, Jierun Chen, Tiezheng Yu, Ruoyu Wang, Yuxin Jiang, Yiming Du, Wendong Xu, Jing Xiong, Taiqiang Wu, Lifeng Shang, Xiao-Hui Li, Ngai Wong, and Haoli Bai. What makes interaction trajectories effective for training terminal agents? *arXiv preprint arXiv:2606.03461*, 2026. URL https://arxiv.org/abs/2606.03461.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025. URL https://arxiv.org/abs/2506.15841.

Yiqi Zhu, Apurva Gandhi, and Graham Neubig. Training versatile coding agents in synthetic environments. *arXiv preprint arXiv:2512.12216*, 2025. URL https://arxiv.org/abs/2512.12216.

A Measurement and Trajectory Construction

A.1 Units and provenance

Each episode has a root objective, persistent workspace, and collection budget. Work orders retain global identities across attempts and context resets. Main and delegated-agent requests are separated by conversation identity; an interruption or retry stays within the same order. Context-history shrinkage is treated as a candidate reset and counted as a real compaction only when an actual summary is observed. The corpus contains 22,810 model requests, 122 main-conversation context segments, and 245 delegated-agent conversations.

Every exported record retains the root task, conversation, segment, order, source-event reference, target type, and token statistics. Supervisor examples additionally identify the observed request, response, and accepted order. Delegated-agent records remain linked to their parent task. Missing reasoning is not synthesized, and internal delegated-agent conversations are not inserted into the parent's visible history. This preserves the distinction between information available to a decision and evidence recovered afterward.

A.2 Action records and length handling

Action records select intervals within a work order and context segment. Their preceding visible interactions remain context; only selected assistant turns become targets. Delegated-agent segments produce separate records. Selection uses a priority-ordered activity classifier (Table 2), whose labels are heuristic rather than correctness judgements. Path novelty follows observed write and edit calls across the episode. Command repetition uses whitespace-normalized strings, and verification-like commands match test, check, build, or benchmark patterns.

Table 2: Priority-ordered action-record retention rules. The first matching rule applies. Categories describe logged activity only; they are retention policy, not judgements of useful progress.

Category	Rule
Exclusion rule	Fewer than 3 valid assistant steps; error or invalid requests exceed half the valid-step count; or at least 5 consecutive identical tool calls with adjacent duplicates comprising at least 30% of calls.
Initial exception	Retain an initial-order interval at full weight after the exclusion checks.
Reduced-weight rule	No logged write; only progress-state writes; fallback with fewer than 15 steps; no new path and over 50% repeated commands; or no new path and final-response similarity above 0.9.
Activity trigger	At least one new path, or at least 3 re-edited paths and 2 verification-like commands.
Remainder	All remaining intervals.

Intervals meeting the exclusion rule supply no targets. The reduced-weight rule assigns turn weight 0.2; the initial exception, activity trigger, and remainder assign 1.0. These describe the existing export policy, not a validated ranking of usefulness. In particular, verification-oriented execution can contain repairs (Section 4); the replay results do not validate these weights. We retain the original weighting for reproducible accounting rather than present it as an empirically supported selection method. System, user, tool-result, inserted-state, and earlier-context messages receive zero weight. Available assistant reasoning, text, tool calls, and end tokens can receive positive weight. We distinguish rendered token volume, positive-weight token count, and fractional-weight mass. These quantities describe the representation and its selection policy, not independent information or measured task value.

All length decisions use the Qwen3.5 tokenizer after chat-template rendering, including tool definitions, with a limit of 262,144 tokens. Recorded reasoning is retained for target turns. Weighted character spans are mapped to token offsets. When a view exceeds the limit, complete oldest zero-weight orders are removed while protecting the leading system and root-objective block. The reduced history uses only model-authored state available before the target interval.

Oversized target intervals are split at complete interactions, preferentially near progress or task-list milestones. Tool calls remain paired with results. The first window preserves the interval's

beginning; later windows use state preceding their cut. Each selected target belongs to one window within a view. At least four assistant turns must fit in the first window and one in subsequent windows; impossible fits are excluded explicitly. These synthetic cuts remain labelled separately because the teacher may have acted with a fuller history.

A.3 HANDOFF RECORDS

Real-summary records pair the actual pre-compaction request with the teacher-authored summary observed after the reset. The fixed runtime preamble is removed, but the summary body is preserved. Its own target text is never used to shorten its input. Four additional history shrinks lack actual summaries and are not counted as real compactions.

Progress records stop their input before a successful state update and use the complete resulting progress text as target. Successful prior writes, edits, or complete read snapshots support reconstruction; failed or incomplete updates do not produce invented targets. An edit fragment is not substituted for the complete resulting state. At most three updates per episode are selected, prioritizing large preceding time gaps.

Real-reset continuation records preserve the context actually visible after compaction. When action records already cover the resumed actions, those records receive a tag instead of being duplicated. Additional continuation records stop at the next order or 40 assistant turns. Of 30 real summary events, 28 have continuations covered by action records, one yields a standalone continuation, and one has no valid continuation response. Summary bodies have a median length of 7,331 characters, ranging from 4,191 to 10,471. Four summary-writing inputs require older context to be reduced.

A.4 CONCRETE DISTINCTIONS IN HANDOFF STATE

A CUDA-task handoff describes the teacher as midway through order 7, then explicitly states that none of that order's work has been executed. It also asks the continuation to repeat order 6's checks only if the solution changes. This distinguishes assigned work from performed work and associates verification with a particular artifact state.

The Horror Lighthouse handoff describes phases A–D as completed and verified, phase E as nearly complete, and the final screenshot set as under review. The governing request preserves project layout, scenario interfaces, trace format, resolution, and input contracts. These are teacher status claims, not independent certification; they illustrate the specific constraints and unresolved checks that continuation must preserve.

Table 3: Information preserved by concrete handoffs. The final column identifies the distinction needed by later execution.

Case	Recorded state	Distinction
CUDA	Current order received; no work executed	Assignment versus action
CUDA	Repeat prior checks only after solution changes	State-dependent validity of evidence
Lighthouse	A–D complete; E nearly done; images under review	Partial versus whole-order completion
Lighthouse	Preserve interfaces and input contracts	New work under existing constraints

A.5 NEXT-ORDER RECORDS

Next-order records pair exact supervisor input messages with decisions accepted by the runtime. Acceptance requires agreement with a persisted non-fallback order, rather than successful transport alone. Initial orders and fallback instructions are excluded. Historical schemas remain unchanged; revised operator labels are not inferred retrospectively.

Full and capped inputs provide alternative views of the same decision. The cap applies 16,000 serialized characters to the workspace-state field, preserving complete paths and current-state prose

when possible, removing whole trailing entries, and marking omissions. Seven inputs require capping and retain all current-state prose. Of 502 inspected supervisor request/response records, 396 are invalid pairs, 15 are associated with fallback orders, and nine do not match the accepted order. The remaining 82 supply paired records.

A.6 Construction diagnostics

The final export contains 572 action records: 327 main-agent and 245 delegated-agent records. They contain 74,288,193 rendered tokens and 26,627,410 positive-weight tokens; the latter comprise 16,984,552 main-agent and 9,642,858 delegated-agent tokens. Fractional-weight mass is 25,854,648.4. Of 561 extraction units, 437 meet the initial exception or activity trigger, 87 the reduced-weight rule, 19 the exclusion rule, and 18 the remainder rule. Those units are order–segment intersections and delegated-agent segments, rather than the 283 audited work orders.

An earlier construction yields 494 action samples and 25.88 million positive-weight tokens. Its later windows usually use the last assistant response as state. The final construction prefers milestones, incorporates task-list state, and reconstructs complete progress text. It splits 29 oversized action intervals into 59 windows. Of 30 later windows, 24 use task-list state, three complete progress writes, two reconstructed edits, and one a last-response fallback. Structured-state coverage rises from 4/29 to 29/30. Several extraction policies and the denominators change, so this comparison is a construction diagnostic rather than an isolated intervention on continuation quality.

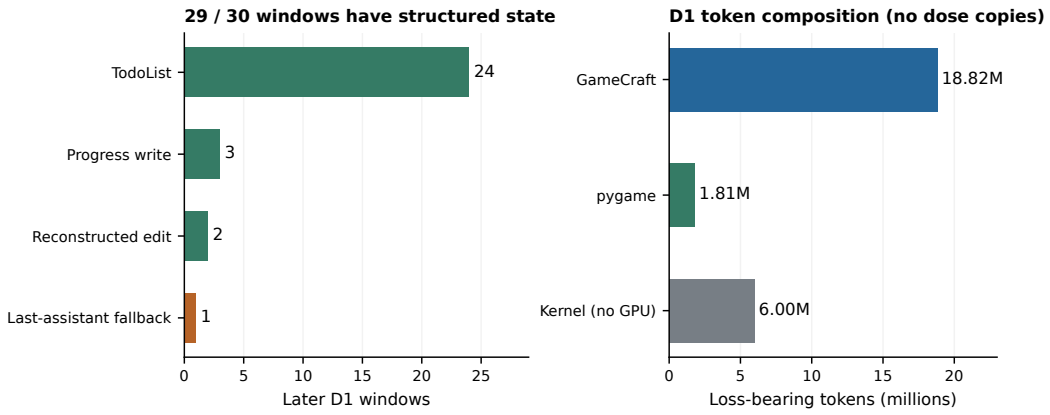

Figure 6: Construction diagnostics: sources of state at later synthetic windows and domain composition of positive-weight action tokens. Structured-source coverage describes retained provenance, not verified summary fidelity.

A narrow exact-retry heuristic sets a retry target to zero weight when its tool name and decoded arguments repeat the preceding failed call exactly. It affects three of 22,763 targeted assistant turns, all following infrastructure interruption: two delegated-agent calls and one screenshot read. These events do not establish that the retries were undesirable. Two additional occurrences in optional prefixes are copies of the same source events.

Optional prefixes select the first valid request/response record at or after a nominal horizon from the first main-conversation segment's initial order, including its response. Both nominal and actual elapsed times are retained. Prefixes overlap longer views; their timestamps can overshoot the nominal cutoff. They provide 241 additional views, not independent tasks or a controlled time-dose study.

The extraction audit checks 1,126 physical records across views for source references, unique per-view identities, weight/mask consistency, assistant-only targets, nonempty targets, token-span totals, and the length cap. Seventy-eight implementation tests cover parsing, classification, reconstruction, rendering, and weighting. These checks establish representation integrity rather than behavioral quality.

Table 4: Optional prefix coverage. Elapsed time is measured from logged timestamps and includes waits.

Nominal hours	Samples	Episodes	Median hours	Maximum hours
1	85	85	1.080	4.457
4	79	79	4.141	9.930
8	77	74	8.278	12.580
24	0	0	—	—

Table 5: Exported views. Rendered tokens include context and repeated prefixes. Capped decisions and prefixes overlap other rows.

View	Samples	Rendered tokens	Positive-weight tokens
Actions	572	74,288,193	26,627,410
Real-summary writing	30	6,766,437	64,616
Progress writing	118	16,496,437	239,791
Extra continuation	1	48,844	362
Next-order decisions	82	1,047,202	119,379
Capped decision alternative	82	1,018,188	119,379
Optional prefixes	241	32,032,262	20,102,331

B Collection Specification and Prospective Evaluation

B.1 Status-gated continuation specification

The revised supervisor receives the root objective, current order, workspace and progress state, prior orders, continuity information, and solver report. The root objective is authoritative; workspace contents and solver reports are observations. A complete judgement requires recorded passing evidence, incomplete denotes unmet checks, and blocked denotes an environmental obstacle without a feasible workaround. Incomplete or blocked work receives a continuation addressing those unmet checks. Completed work can receive one extension selected from Table 6, informed by observed weaknesses and previous orders.

Table 6: Extension families in the revised specification. These categories postdate the audited collection.

Family	Required change and evidence
Depth	Make behavior depend on a new upstream capability, exercised end to end.
Width	Add a jointly satisfied requirement while retaining earlier checks.
Scale	Exercise larger size, duration, or load against a measured target.
Robustness	Handle specified malformed, boundary, adversarial, or concurrent inputs.
Fidelity	Close a measurable gap against a reference or specification.
Integration	Connect existing subsystems and check the combined flow.

The new-check requirement applies to extensions, while continuations may establish missing evidence with a previously unmet check. Pure documentation and repeated audits do not by themselves qualify as behavior-changing extensions. Each order defines a finishable inspect–implement–run–repair cycle; instructions cannot impose a minimum duration or require idling. Recovery resumes the same unfinished order, and any fallback retains its generation-failure source.

The parser enforces identity, predecessor links, status/action consistency, nonempty instructions and acceptance checks, and verbatim repetition checks. It does not execute acceptance checks or enforce semantic evidence sufficiency. The revised prompt and solver policy postdate the corpus. A focused implementation suite reports 57 passing tests and one skip; the specification has no controlled outcome evaluation here.

B.2 Artifact and continuation comparisons

A prospective outcome study should fix root-task checks before evaluating later work. At saved boundaries, compare additions, repairs, regressions, and unavailable checks with supporting execution evidence. Games require interaction rather than screenshots alone. Independent checks should supplement the supervisor's own acceptance criteria. Missing historical states should remain unavailable comparisons rather than inferred successes.

Compare status-gated continuation with generic same-session improvement, predetermined extensions, and a longer initial instruction on shared starting workspaces. Hold solver, tools, prerequisites, and resource budgets fixed. Separate controls matched by requirement family and order count from unrestricted adaptive selection, which also changes family choice and timing. Charge supervisor generation and regression checking to the budget, including failures. These comparisons distinguish gains from additional scope, adaptation, and selection.

B.3 State, resumption, and student assessment

At common saved workspaces, cross summary producers with continuation policies and include fixed-summary and intact-context controls. Assess completion, repeated work, forgotten constraints, and unsupported state claims. Compare summaries with prior trace evidence as well as their utility in subsequent execution. Real resets and synthetic cuts require separate reporting because they provide different evidence about what the teacher could see.

The action, handoff, and next-order views can support student assessment of execution, evidence-grounded state writing, and later-order selection. All views of a root task, including delegated-agent records, prefixes, and alternative contexts, must remain in one split. The current export has no instantiated held-out split or reported student checkpoint. Qwen3.5-35B-A3B chat formatting describes representation compatibility; it does not establish multimodal handling or behavioral capability. Downstream assessment should report source coverage and execute task checks.

Duration studies should hold prompts, session policy, services, and prerequisites fixed, distinguish active interaction from waiting, and use prefix-local evidence on shared eligible tasks. State practical effect margins in advance, retain interrupted runs, and cluster uncertainty by root task. These are prospective comparisons, distinct from the retrospective analyses reported here.

C Instruction Review and Replay Details

C.1 Full-text review

All 61 extension targets retain the entire instruction and every acceptance entry. Two isolated contexts of the same judge model each rate every target, without access to initial ratings, the manuscript, or each other's results. Mandatory repairs of specified defects qualify as required behavioral change; repairs conditional on discovering a defect do not suffice. Each output includes a label, a reason of at most 30 words, and a verbatim evidence quote of at most 45 words. Validation checks exact identifier coverage, allowed labels, lengths, and presence of every quote in its input.

Both runs yield 24 yes and 37 no labels, agreeing on every item. The single change from the earlier clipped review concerns mandatory architecture and language-standard settings in a delivered extension loader. Treating this build-behavior case as negative yields 23/61. Full-text review changes both context availability and criterion focus relative to the initial study, so the label change cannot be attributed solely to removing clipping.

C.2 Initial broader review

The initial review covers all 82 accepted non-fallback orders. Three isolated judge contexts each receive two of six cyclically assigned batches, with batch sizes 14, 14, 14, 14, 13, and 13. Each item receives one judgement per criterion at high reasoning effort. Judges see the rubric before rating and are instructed to express uncertainty rather than target an aggregate result.

The criteria ask whether preceding status is supported, the instruction is grounded in observed state, acceptance checks are checkable, extensions require functional change, and work is feasible under the stated environment. Labels are yes/no/unclear/not-applicable with reasons of at most 20 words. The required-change criterion is inapplicable to continuations.

Inputs retain at most 900 characters of the root objective, 1,500 of the preceding order, 2,800 initial plus 900 final characters each of workspace state and solver report, 500 of continuity information, and 300 of retry suffix. Instructions retain 4,000 characters and acceptance lists 2,500. Omission markers identify clipping. All 82 evidence bundles and 58 target orders contain omissions.

Table 7: Initial clipped-review results, separate from the full-text replication.

Criterion	Yes	No	Unclear	N/A
Preceding status supported	61	20	1	0
Instruction grounded in observed state	82	0	0	0
Acceptance criteria checkable	82	0	0	0
Extension requires functional change	23	38	0	21
Feasible in the stated environment	79	2	1	0

All 20 status-support negatives cite explanatory responses conflicting with code-only delivery constraints. Their distribution across contexts is 10/28, 0/28, and 10/26; the middle context emphasizes preceding-order completion over unresolved root requirements. Items differ across contexts, so this is scope sensitivity rather than a measured inter-rater disagreement rate. These ratings are not a calibrated supervisor error estimate.

C.3 Patch selection and authentication

All six pygame episodes were inspected, with one later-order repair selected per root task. Selection was retrospective and purposive. Available final source, recoverable complete edit arguments, and an executable diagnostic were required; an exhaustive candidate screening count was not recorded. Table 8 retains the mapping between selected actions and their source orders. Event indices are zero-based physical trace lines.

Table 8: Source-event provenance for the six selected repairs.

Game	Source event	Order index	Instruction category
Space Battle	202	2	Verification/conditional repair
Imps & Assassin	207	1	Fallback
Dr E's Lab	198	2	Verification/conditional repair
Shadow Vale	154	1	Mandatory behavioral change
Grid RPG	66	1	Verification/conditional repair
Green Ninja	153	1	Verification/conditional repair

Authentication recomputes the SHA-256 signature of the complete serialized edit arguments and verifies that the replacement block occurs exactly once in final source. Reversing that replacement produces the removed-patch variant in memory, leaving the archived artifact intact. Grid RPG and Green Ninja require recovery from full source traces because parsed argument previews are bounded. Both recovered edits match the recorded signatures. Final-source, counterfactual-source, and replay-implementation hashes are retained with the results.

C.4 Replay environment and stimuli

Replays use Python 3.11.6, pygame 2.6.1, SDL 2.28.4, and dummy video/audio drivers. The Imps & Assassin archived suite uses pytest 8.3.5. The first two game experiments exercise world updates and rendering; Dr E's Lab and Shadow Vale exercise actual artifact components; Grid RPG uses its archived event/clock harness to drive the main loop; Green Ninja uses archived whole-game physics probes.

Space Battle. The patch clears held and pressed keys when starting a match and returning to the menu. Seeds are 101, 202, 303, 404, and 505; held keys are W, S, Up, and Down. The target navigation key remains held through countdown; the opposite key restores the two-player selection and is released, then Enter starts the match and is released. At a fixed 60 Hz update rate, maximum ship speed is recorded for 20 gameplay frames. Speed above 1 px/s indicates inherited thrust. Each fresh-input control uses a new instance, starts the match with no held keys, verifies a zero-motion baseline over 20 idle frames, then checks that a fresh press increases speed by more than 1 px/s over 20 frames. All 86 archived tests pass with no skips.

Imps & Assassin. The patch guards enemy scheduling after game over. Four initial spawner phases—idle, announce, active, and cleared—are crossed with the five seeds above. Lethal damage invokes the ordinary death path, followed by 120 fixed-step world updates and renders. New-wave or portal scheduling events are counted. Separate live-player instances supply matched ordinary-scheduling controls. Removal produces post-death scheduling in all 20 conditions; retention produces none, while both variants schedule normally in all live-player controls. All 31 archived tests pass without skips.

Dr E's Lab. The patch substitutes exponential for Euler damping of collectible scatter velocity. Initial speeds are 50, 100, 150, and 200 px/s; directions are 0, 30, 60, 120, and 180 degrees. Each condition advances the pickup for 0.2 seconds in 6, 12, and 24 steps, without walls and with the player outside magnetic range. A final-velocity-vector spread greater than 10^{-8} px/s violates partition invariance. Both variants must retain nonzero displacement and nonzero decaying velocity. The probe concerns velocity, not position-integrator invariance.

Shadow Vale. The patch lowers a sleeping bat from tile row 8 to row 9. The actual level, enemy factory, player rectangle, and coordinate conversion define player–bat geometry. Ground-row-15 approaches use horizontal offsets $-60, -30, 0, 30, 60$ px and update intervals $1/120, 1/60, 1/30, 0.1$ seconds. A row-12 nearby-platform approach supplies the control. The wake radius remains 175 px. These conditions test activation on a specific ground route, not whole-level reachability.

Grid RPG. The patch processes a movement press before its release can remove it from the held-key list. The five fixed seeds are crossed with frame times of 6.94, 16.7, 33.3, and 100 ms. An adjacent walkable tile without a hero or enemy is verified before each trial. A press/release pair in one event batch must produce exactly one turn and a one-tile displacement. Separate-frame events supply the control. The experiment does not evaluate long held-key sequences.

Green Ninja. The patch introduces physics substeps of at most $1/60$ second. The 28 archived tunneling probes combine six projectile phase offsets and eight thin-ledge offsets with $1/20$ and $1/15$ second update intervals. Six ordinary $1/60$ second projectile probes supply controls. Removal fails projectile phases 3 and 4 and ledge phase 7 at the larger interval; retention passes all probes. The test set is a targeted subset of the archived graphical checks.

C.5 Archived visual context

The saved Space Battle capture and 17-case image-inspection log predate the selected input repair. They document the surrounding artifact, while the separate patch replay establishes the input-state effect. Their solver-authored image checks are not attributed to the later order or treated as independent game-quality scores.

The archived capture is shown in Figure 5 alongside the main-text behavioral analysis.

D Additional Corpus and Usage Accounting

D.1 Order-level accounting

Initial orders have medians of 120 assistant steps, 19 newly logged paths, 33 verification-like commands, and 23.6 hours. Orders at indices 1 and 2 have medians of 24 steps, one new path, and eight verification-like commands, with median residence times of 3.3 and 2.5 hours. Index 3+ orders have

medians of 11 steps, zero new paths, three verification-like commands, and 1.5 hours; 72 of 120 add no newly logged path. Residence times include waits and interruptions. Logged write/edit paths omit shell-generated files and are activity proxies.

Table 9: Later orders by domain. Reached later work counts episodes with a later order; zero-new counts orders without a newly logged write/edit path.

Domain	Reached later work	Orders	Median steps	Zero-new
GameCraft	12/58	17	45	35.3%
pygame	6/6	14	53	21.4%
Other kernels (no GPU)	12/18	39	22	28.2%
CUDA language-model tasks (no GPU)	11/11	120	11	62.5%

Main-conversation segments have a median of 136 assistant turns and a maximum of 322. Their final prompt sizes have a median of 222,584 teacher-tokenizer tokens, with 51 of 122 exceeding 256,000. These measure visible context at segment ends rather than API usage or student-format fit.

D.2 Recorded usage and coverage

Usage fields are taken from logged model responses. Validation checks unique case/request identities and source events, nonnegative counts, and equality of input plus output to reported totals. Usage is available for 22,759 of 22,810 requests (99.8%); the 51 missing records are not imputed. Main-agent and delegated-agent requests are included, while separate supervisor calls and the 48-hour collection are excluded.

Table 10: Whole-episode solver usage. The request column counts records with usage; input tokens include repeated context.

Episode group	Requests	Input tokens	Output tokens
Hardware-eligible games	17,224	1,939,787,010	18,783,042
GPU-unavailable tasks	5,535	874,468,300	5,741,059
All 93 episodes	22,759	2,814,255,310	24,524,101

Explicit cached-token counts are present in 21,922 records and sum to 1,645,773,568; 837 otherwise usable records omit this field. Cached tokens are a subset of input, and the 14,096,719 reported reasoning tokens are a subset of output. Neither is added a second time. Provider-reported totals are not a monetary cost estimate or a measure of active compute.

D.3 Separate 48-hour collection

A separate collection used 48-hour budgets without the session fuse. Of 87 intended tasks—59 GameCraft and 28 ProgramBench—86 have usable episodes: 58 GameCraft and all 28 ProgramBench. ProgramBench requires reimplementing compiled command-line programs. Collection summaries report 82 budget-reaching runs and, separately, 82 single-attempt runs; these sets have not been shown to coincide. Four usable episodes end early, after 29–48 hours, because of collection failures.

Among 57 GameCraft tasks shared with the 24-hour corpus, the summary reports 12 entering the supervisor loop in the earlier configuration and three leaving the initial attempt in the later configuration, with no task in common. These events are not equivalent: an attempt can end in a retry of the same order. Launch-state records were located, but their referenced historical traces were unavailable for reconstruction. Prompts and services also changed. This collection therefore remains descriptive appendix context and contributes to none of the main-text order, usage, review, or export totals.