SWE-SHIFT: EVALUATING CODING AGENTS IN EVOLVING WORKSPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Repository-level coding benchmarks largely assume that tasks and environments remain unchanged throughout execution. In software development, code, dependencies, runtime state, and requirements can change while an agent is still working. We introduce SWE-Shift, a benchmark for evaluating coding agents under such mid-execution changes. From the same saved history and workspace, a Clean continuation proceeds without intervention, while a matched Shifted continuation receives a task-relevant update. Success requires completing the current task, preserving still-valid work, and satisfying the applicable state and evidence checks. The benchmark contains 120 scenarios across 12 change families, each evaluated at nine scheduled intervention positions. We evaluate 17 coding-agent systems, all of which have lower Shifted than Clean success. GPT-6 Astra drops from 67.35% to 24.79%, with 20.34% paired success. Tool/session changes and stale evidence yield zero paired success across all systems. Late-position paired success is lower for some systems and higher for others. These results show that success on the unchanged task can coexist with failures to preserve work, recover state, or refresh evidence after an update.

1 INTRODUCTION

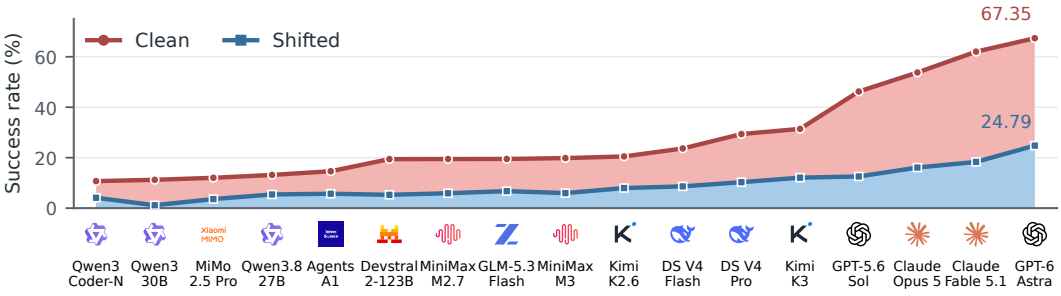


Figure 1: **Success drops after workspace changes.** Clean (red) and Shifted (blue) success rates across 17 coding-agent systems, ordered by Clean success.

Repository-level benchmarks such as SWE-bench (Jimenez et al., 2024) have made it possible to evaluate coding agents on realistic software engineering tasks rather than isolated code generation. An agent must understand an issue, navigate a repository, modify executable code, and satisfy tests. Yet these benchmarks largely assume that once execution begins, the task and its environment remain unchanged. Real software development is less static. Code may be edited, dependencies may change, runtime state may evolve, and requirements or available information may be updated while an agent is still working.

This raises a different reliability question:

Can a coding agent continue to succeed when task-relevant changes occur during execution?

This question is not answered by evaluating agents on changed tasks alone. If an agent fails after a change, we do not know whether it was unable to solve the underlying task in the first place or

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.

whether the change disrupted an execution that would otherwise have succeeded. A useful evaluation must therefore separate these two sources of failure.

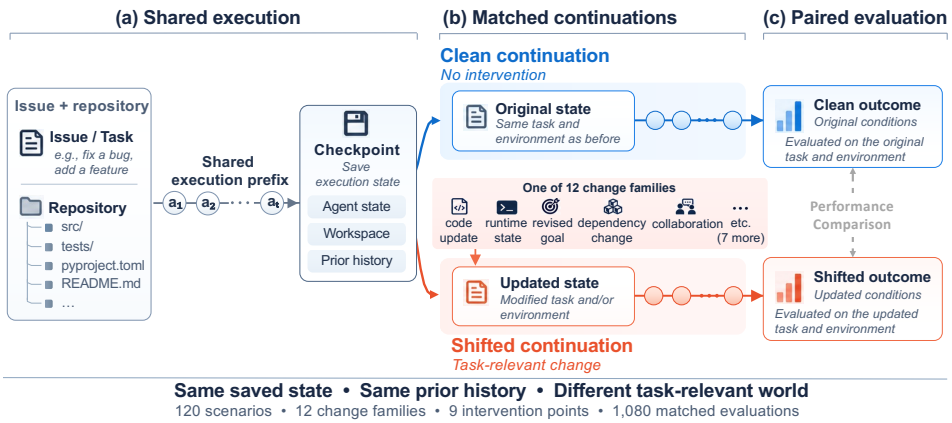


Figure 2: **Per-system evaluation pipeline of SWE-Shift.** From the same saved execution state, the agent resumes in two matched continuations: Clean proceeds without intervention, while Shifted introduces a task-relevant change. Outcomes are evaluated under the original and updated conditions, respectively.

We introduce SWE-Shift, which addresses this problem through matched Clean and Shifted continuations (Figure 2). The agent first works on the task normally. At an intermediate point, we save its execution state and continue from the same prefix in two ways: the Clean continuation proceeds without a change, while the Shifted continuation introduces a task-relevant change. Because the two continuations share the same prior actions, workspace, and observable history, Clean provides a matched baseline for whether the agent can succeed from that state, while Shifted tests whether that success survives the change.

Constructing such an evaluation also requires the changes themselves to be meaningful. A change should matter to the task, but it should not make the task impossible. SWE-Shift therefore includes scenarios in which ignoring the change can invalidate an otherwise successful continuation, while a correct continuation remains possible after the change. We also introduce each change at multiple points during execution, rather than tying the benchmark to a single moment in the agent's trajectory. Finally, success is judged against the requirements that apply after the update, including preservation and evidence requirements that the original task tests may not cover.

SWE-Shift contains 120 scenarios across 12 change families, covering code and dependency updates, collaboration, runtime and session state, external effects, revised goals, and stale or misbound evidence. Nine intervention positions per scenario yield 1,080 scheduled evaluation cells per system.

Our evaluation of current coding agents reveals a substantial gap between success in unchanged and changing environments. As shown in Figure 1, all 17 evaluated coding-agent systems have lower observed success after the intervention. GPT-6 Astra, which has the highest Clean success, drops from 67.35% on Clean continuations to 24.79% on Shifted continuations. The size of the gap also varies across change families and intervention positions, suggesting that robustness to mid-execution change captures behavior that static task success alone does not reveal.

Our contributions are summarized as follows:

- We identify **robustness to mid-execution change** as an evaluation target beyond static task success, distinguishing failures on the underlying task from losses of successful continuations after a task-relevant change.
- We introduce SWE-Shift, a matched-continuation benchmark with 120 scenarios across 12 change families and nine intervention positions, enabling controlled comparison of Clean and Shifted executions from the same intermediate agent state.
- We evaluate 17 coding-agent systems and find a substantial gap between Clean and Shifted success, with failures varying across change families and intervention positions. Recorded trajectories and diagnostic controls further characterize these failures.

2 Related Work

Repository benchmarks such as SWE-bench evaluate issue resolution but normally keep the task and environment fixed during execution (Jimenez et al., 2024). Related dynamic coding evaluations study different kinds of change: SWE-Touch injects validated user edits during execution, SyncMind begins recovery from repository inconsistencies, SWE-INTERACT reveals or revises requirements through dialogue, and SpecPath varies specification history before implementation (Tan et al., 2026; Guo et al., 2025; Raghavendra et al., 2026; Wu et al., 2026a). ToolSandbox contributes state dependencies, milestone/Minefield trajectory evaluation, while Gaia2 adds independently evolving asynchronous events and action-level write verification (Lu et al., 2025; Froger et al., 2026). Table 1 compares these evaluation designs. "Mid-episode" denotes an external intervention after execution begins. "Fresh evidence" means that completion claims are checked against the current state. SWE-Shift combines three controls: matched continuations from the same recorded prefix, changes to multiple parts of the task environment, and checks that reject the loss of still-valid behavior even when the original task tests pass. Table 7 in Appendix E provides a detailed comparison.

Table 1: **Benchmark design at a glance.**

Benchmark	Mid-episode	Same prefix	Cross-plane	Preserve/harm	Multi-position	Trajectory oracle	Fresh evidence
SpecPath	×	×	×	×	×	×	×
SyncMind	×	×	×	×	×	×	×
SWE-INTERACT	✓	×	×	×	×	×	×
ToolSandbox	×	×	×	✓	×	✓	×
SWE-Touch	✓	×	×	✓	✓	×	×
Gaia2	✓	×	✓	✓	×	✓	×
SWE-Shift (Ours)	✓	✓	✓	✓	✓	✓	✓

3 SWE-Shift Benchmark

We first describe how task-relevant changes are designed (Section 3.1). We then explain how matched continuations and intervention timing support controlled comparisons (Section 3.2). Finally, we define the success criteria and metrics used to evaluate these continuations (Section 3.3).

3.1 Designing Task-Relevant Changes

SWE-Shift builds 120 change scenarios on 23 task-root snapshots from 17 repositories, drawn from SWE-bench Verified (Chowdhury et al., 2024), Multi-SWE-bench (Zan et al., 2025), and SWE-bench Multilingual (Khandpur et al., 2025). Each scenario combines an existing repository task with a task-relevant change introduced during execution. Table 4 in Appendix C.1 provides the scenario counts by source.

To evaluate adaptation, these changes must challenge the agent without making successful completion impossible. We therefore require each change to be both *relevant* and *recoverable*. Relevance means that ignoring the change can invalidate an otherwise successful continuation. Recoverability means that, after the change, at least one valid way to complete the task remains available through the agent-accessible tools. Together, these requirements exclude changes that do not matter to the task and changes that make the task unsolvable.

To turn such a change into an executable evaluation scenario, we specify four elements: the initial repository task, an authorized mid-execution change, a way for the agent to encounter or discover that change, and executable checks for a correct continuation. These checks reflect the requirements that apply after the update: some original requirements remain in force, while others may be added or withdrawn. A correct continuation must satisfy the current requirements while preserving work and effects that remain valid.

Using this scenario structure, SWE-Shift covers 12 change families, with ten scenarios in each family. Figure 3 organizes them into three broad groups according to what the change affects. F01–F07 modify the executable world or its dependencies, including user and collaborator edits, interface changes, runtime and session state, and external effects. F08–F10 change what the agent is required to act on through newly disclosed constraints, corrective feedback, or an authorized revision of the goal. F11–F12 change the validity or binding of evidence: an observation may describe an earlier state or refer to the wrong request, branch, or session.

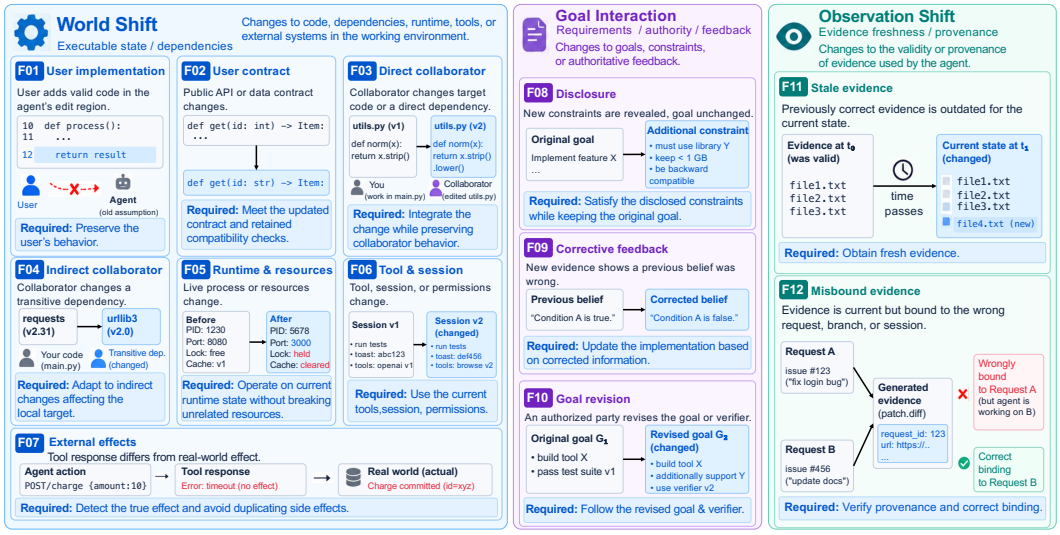


Figure 3: **What SWE-Shift evaluates.** The 12 change families are organized into three broad groups. World Shift (F01–F07) tests changes to executable state or dependencies. Goal Interaction (F08–F10) tests newly disclosed requirements, corrective feedback, and revised goals. Observation Shift (F11–F12) tests stale or misbound evidence. Each panel illustrates the intervention and the required continuation.

3.2 Matched Continuation Protocol

SWE-Shift compares continuations with and without a mid-execution change from the same prior execution (Figure 2). For each system, scenario, and intervention position, the agent first executes a single shared prefix. At the scheduled checkpoint, the runner saves the agent-visible history, workspace, controller state, available tools, system settings, and execution counters. The saved state is then restored twice to create a matched pair of continuations.

In the **Clean** continuation, execution proceeds without the scenario's task-relevant update. In the **Shifted** continuation, the authorized update is applied at the fork. Up to that point, the two continuations have the same recorded execution history and workspace state. Only after the intervention can their observations and actions diverge. This differs from comparing independently generated runs, where differences before the change may already affect the outcome. The Clean continuation therefore provides a matched baseline for whether the agent can complete the task from that particular intermediate state, while the Shifted continuation tests whether that success survives the change.

The update is introduced through the scenario's declared interface. Depending on the case, the agent may encounter it as a workspace change, a delivered message, a changed tool or session response, or new information returned by an allowed state query. The evaluator may use additional held-out checks to determine correctness, but these checks do not provide additional information to the agent. Thus, the paired comparison changes the task-relevant conditions after the fork while keeping the observable pre-intervention execution fixed.

3.2.1 Intervention Timing

A single intervention point could make the evaluation depend strongly on when the change happens to occur. SWE-Shift therefore evaluates every scenario at nine progressively later intervention positions, denoted T10–T90. These positions are constructed from an eligible prefix of a reference solution, ending before the first action whose correctness depends on the update. The construction procedure provides nine distinct intervention anchors spanning 10%–90% of the eligible reference progress for every admitted scenario.

During model evaluation, the intervention position determines how many agent actions are executed before the fork. The evaluated agent is not required to follow the reference trajectory used to construct the schedule. Consequently, T10–T90 represent progressively later *scheduled positions*, rather than normalized semantic progress through the task. Different systems may therefore have reached different workspace states when the same nominal intervention position is triggered. If an agent terminates before the scheduled intervention or changes the state in a way that affects whether the update can still be applied, the cell retains its assigned intervention position and is handled by the benchmark's outcome protocol without rescheduling.

3.3 Success Criteria and Metrics

A continuation is counted as successful only if it satisfies every check that applies under the conditions in which it ends. For the Clean continuation, these are the original task and environment conditions. For the Shifted continuation, they are the conditions after the update. The original task tests are therefore only one part of the evaluation. Depending on the scenario, success may additionally require preserving still-valid behavior, following the current goal or interface, avoiding prohibited effects, or obtaining evidence that is valid for the current state. The implementation groups these requirements into task, preservation, recovery, authority, effect, evidence, budget, and terminal-integrity checks. The criterion is conjunctive: failing any applicable check makes the continuation unsuccessful. Table 6 summarizes the required evidence.

For model–agent system m, scenario i, and intervention position t, let $C_{mit} \in \{0,1\}$ and $S_{mit} \in \{0,1\}$ denote verified success in the Clean and Shifted continuations, respectively. Ordinary paired credit requires success in both continuations:

$$R_{mit} = C_{mit} S_{mit}. \tag{1}$$

This joint-success component measures success in both observed continuations: under the original conditions and after the task-relevant change.

Some agents complete the required task before the scheduled intervention is reached. Let $P_{mit} \in \{0,1\}$ denote verified pre-intervention completion under the benchmark's early-completion procedure. The score assigned to an intervention cell is

$$K_{mit} = \max\{R_{mit}, P_{mit}\}. \tag{2}$$

Thus, a cell receives credit either because both matched continuations succeed or because the task was already completed before the agent could be exposed to the change.

We aggregate scenarios hierarchically: each of the 12 change families receives equal weight, followed by equal weight for clusters within a family and scenarios within a cluster. Let w_i denote the resulting normalized scenario weight, and let $\mathcal{T} = \{\mathrm{T10}, \ldots, \mathrm{T90}\}$ denote the nine intervention positions. We report

$$\mathrm{Clean}_m = \frac{1}{9} \sum_{\substack{t \in \mathcal{T} \\ i}} w_i C_{mit}, \quad (3) \qquad \mathrm{Shifted}_m = \frac{1}{9} \sum_{\substack{t \in \mathcal{T} \\ i}} w_i S_{mit}, \quad (4) \qquad \mathrm{PSR}_m = \frac{1}{9} \sum_{\substack{t \in \mathcal{T} \\ i}} w_i K_{mit}. \quad (5)$$

Our primary paired metric is the **Paired Success Rate (PSR)**. PSR averages credit for joint success or verified completion before intervention.

We additionally report the **Conditional Retention Rate (CRR)**, which asks what fraction of Clean success is also retained in the Shifted continuation:

$$\mathrm{CRR}_m = \frac{\sum_{t \in \mathcal{T}} \sum_i w_i R_{mit}}{\sum_{t \in \mathcal{T}} \sum_i w_i C_{mit}}, \tag{6}$$

when the denominator is nonzero. Its numerator is joint success R and excludes the early-completion credit in PSR. Clean, Shifted, PSR, and CRR therefore summarize baseline continuation success, success under changed conditions, paired success, and retention, respectively.

4 Experiments

Our experiments ask whether coding capability survives a workspace change (RQ1), which changes expose shared or system-specific weaknesses (RQ2), whether their timing matters (RQ3), and what

a failed continuation can hide from ordinary task tests (RQ4). We then examine how state handling and post-fork token budgets affect success.

4.1 Experimental setup

We evaluate 17 coding-agent systems, listed in Table 8, using the same 120 scenarios and nine intervention positions, with the same available tools and post-fork token budget $B = 1{,}000{,}000$ tokens per continuation, counted as cumulative input plus output tokens across all post-fork model calls. Each row in Table 2 denotes a deployed model–agent system. Provider-hosted systems use their native agent applications: Codex app for GPT-5.6 Sol and GPT-6 Astra, and Claude Code app for Claude Opus 5 and Claude Fable 5.1, all at `xhigh` reasoning. The open-weight coding-agent systems were served locally on four compute nodes with eight NVIDIA B300 GPUs each (32 GPUs total).

Clean, Shifted, PSR, and CRR use the family–cluster–scenario weighting in Section 3. The reported main results are point values.

4.2 RQ1: Does coding capability survive a workspace change?

Workspace changes reduce continuation success. Success drops in all 17 Clean–Shifted comparisons. Across the 17 coding-agent systems in Table 2, Clean success ranges from 10.71% to 67.35%, compared with 1.16–24.79% for Shifted and 0.90–20.34% for PSR. GPT-6 Astra attains 67.35% Clean, 24.79% Shifted, and 20.34% PSR. Its CRR of 30.20% means that fewer than one-third of its weighted Clean successes also succeed under change.

Table 2: **Continuation performance (%).** Full identities appear in Table 8.

System	Cln. ↑	Shf. ↑	PSR ↑	CRR ↑
GPT-6 Astra (2026b)	67.35	24.79	20.34	30.20
Claude Fable 5.1 (2026a)	62.01	18.33	16.88	26.02
Claude Opus 5 (2026b)	53.76	16.09	14.78	26.54
GPT-5.6 Sol (2026a)	46.22	12.56	11.30	23.94
Kimi-K3 (2026b)	31.41	12.09	8.20	25.12
DeepSeek V4-Pro (2026b)	29.37	10.29	8.83	29.15
DeepSeek V4-Flash (2026a)	23.65	8.63	7.60	30.99
Kimi-K2.6 (2026a)	20.50	7.97	7.10	34.63
MiniMax-M3 (2026b)	19.84	5.97	6.20	29.08
GLM-5.3-Flash (2026)	19.50	6.75	5.30	25.18
MiniMax-M2.7 (2026a)	19.47	5.91	5.70	27.84
Devstral 2-123B (2025)	19.44	5.29	4.70	22.07
Agents-A1 (2026)	14.63	5.72	4.50	29.01
Qwen3.8-27B (2026a)	13.17	5.43	4.34	30.45
MiMo V2.5-Pro (2026)	12.03	3.61	2.60	21.26
Qwen3-30B-A3B (2025)	11.23	1.16	0.90	6.02
Qwen3-Coder-Next (2026b)	10.71	4.12	2.10	14.97

Higher Clean success does not imply higher retention. In Table 2, Kimi-K3 attains 31.41% Clean success, 8.20% PSR, and 25.12% CRR. DeepSeek-V4-Pro has lower Clean success (29.37%) yet higher PSR (8.83%) and CRR (29.15%). Kimi-K2.6 reaches 34.63% CRR despite a lower Clean score of 20.50%. CRR measures retention conditional on Clean success and complements the absolute continuation scores.

4.3 RQ2: Which changes are persistent failure points?

Tool/session changes and stale evidence expose shared weaknesses. Under the common evaluation setting, Figure 4(a) shows the following family-level PSR results:

1. **Tool and session changes (F06).** A tool, authorization, schema, or session protocol changes after execution begins, invalidating earlier calls. The agent must discover and use the current permitted interface while preserving unrelated session state. PSR is 0.00% for 17 of 17 systems.
2. **Stale evidence (F11).** Evidence refers to an outdated state. The agent must obtain evidence bound to the current state before editing or accepting completion. PSR is 0.00% for 17 of 17 systems.
3. **Runtime and resource changes (F05).** A process, lock, port, cache, or resource owner changes while the task is in progress. The agent must rebind to the live state and recover without harming unrelated resources. MiMo-V2.5-Pro obtains 0.60% PSR. 16 of 17 systems score zero.

4. **Disclosure changes (F08).** A previously hidden constraint or fact becomes visible during execution. 13 of 17 systems score 0.00% PSR. Claude Fable 5.1 reaches 16.20%, while GPT-6 Astra reaches 14.90%.

The remaining families reveal selective strengths. In Figure 4(a), Kimi-K3 reaches 63.50% on F01 (User implementation), while Claude Fable 5.1 reaches 60.80%. On F04 (Indirect collaborator), GPT-6 Astra scores 57.30% and GPT-5.6 Sol scores 39.30%. On F07 (External effects), GPT-6 Astra reaches 22.10% and GLM-5.3-Flash reaches 18.50%. 13 of 17 systems score zero. A higher overall PSR does not imply higher PSR in every change family.

4.4 RQ3: Does the timing of a change affect continuation?

Late-position PSR can be lower or higher. Figure 4(b) shows PSR at the nine scheduled intervention positions. Comparing mean PSR at T70–T90 with T10–T30, GPT-6 Astra loses 20.05 percentage points, Claude Fable 5.1 loses 15.43, GPT-5.6 Sol loses 12.53, and Claude Opus 5 loses 11.93 points. GLM-5.3-Flash instead gains 2.97 points, Kimi-K2.6 gains 2.20 points, and DeepSeek-V4-Flash gains 1.60 points.

Timing profiles differ across systems. The timing profiles in Figure 4(b) show that mean PSR at T70–T90 is lower than at T10–T30 for both GPT systems, both Claude systems, and both MiniMax systems. The corresponding mean is higher for GLM-5.3-Flash, both DeepSeek systems, and both Kimi systems. These comparisons summarize scheduled positions within each system's trajectory.

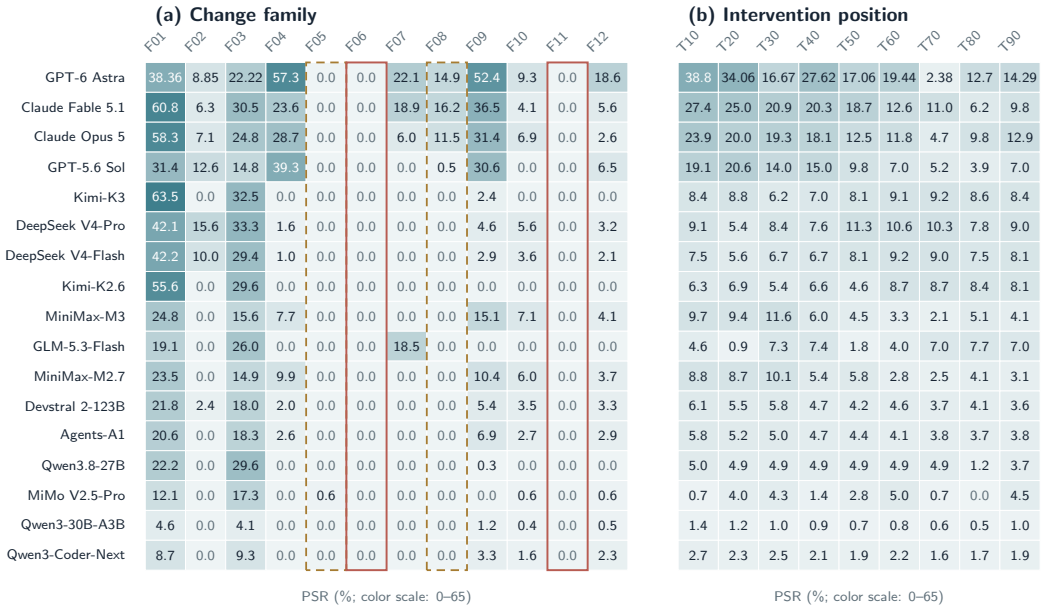


Figure 4: **Paired success across changes and intervention positions.** PSR by change family (a) and intervention position (b). Solid coral outlines highlight tool/session changes (F06) and stale evidence (F11). Dashed gold outlines highlight runtime/resource changes (F05) and disclosure (F08).

4.5 RQ4: What do task tests miss after an update?

After an update, passing the original tests or completing one corrective action may still leave other requirements unmet. We examine three cases from the 24-case qualitative audit set (Appendix G), using the recorded GPT-5.6 Sol actions and evaluator results to explain the final outcomes.

Tests can pass after a collaborator's change is removed. In the Django case, a collaborator adds an API that controls when migration operations can be combined. The agent initially uses this API,

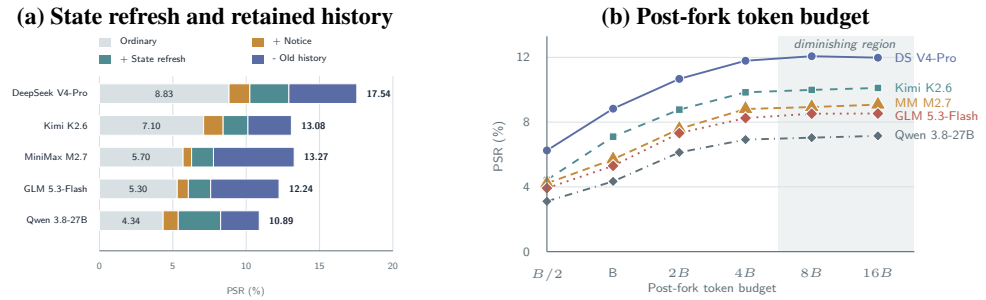


Figure 5: **Effects of state handling and post-fork token budget on PSR.** (a) Bars show cumulative state-access interventions from the Table 2 baseline to the final PSR. (b) Curves show PSR across post-fork token budgets. B is the baseline of 1,000,000 input plus output tokens per continuation, and the gray region marks $8B$–$16B$.

then removes it while fixing the original issue. All 39 original tests pass, but the preservation check reports that the collaborator's API is missing.

Compensation can succeed while workspace checks fail. In the Astropy case, a cancellation arrives after an external effect has already occurred. The agent compensates for that effect, and the tool record confirms that compensation succeeded. The workspace-preservation check still fails.

The final edits need fresh evidence. In the scikit-learn case, the agent restores access to current task evidence, then edits the code without querying that source again. The original task checks pass, but the required fresh observation is missing. The patch also exceeds the allowed edit scope.

SWE-Shift therefore evaluates both the final workspace and the recorded actions against the requirements that apply after the update.

4.6 Diagnostic Controls

Figure 5 shows how state access, retained history, and post-fork token budget affect PSR in five systems, using the corresponding PSR values in Table 2 as baselines.

4.6.1 State Refresh and Retained History

Adding history reset yields the largest incremental gain in four of five systems. In the state-access control (Figure 5(a)), we hold tasks, obligations, workspaces, tools, and the post-fork token budget fixed. Starting from the PSR baseline in Table 2 (Ordinary), we cumulatively add a change notice, a read-only state refresh, and removal of pre-change dialogue. Adding history reset contributes 5.50 points for MiniMax-M2.7. Qwen3.8-27B gains more from state refresh (2.89 points) than from history reset (2.63). The complete intervention adds 5.98–8.71 PSR points, but final PSR remains 10.89–17.54%. The increments measure each addition to the preceding intervention setting.

4.6.2 Post-fork Token Budget

Gains diminish beyond $4B$. In a separate control (Figure 5(b)), we test sensitivity to the post-fork token budget, holding tasks, updates, prefixes, tools, decoding, and scorer fixed. We scale the budget for each continuation from $B/2$ to $16B$, where $B = 1{,}000{,}000$ cumulative input plus output tokens. This tests 500,000–16,000,000 tokens per continuation. The curves report absolute PSR across budgets. The B point is the same baseline reported in Table 2. Each doubling up to $4B$ adds 0.79–2.67 points across the five systems. By contrast, $4B \to 8B$ adds 0.12–0.28 points, and $8B \to 16B$ changes PSR by -0.09 to $+0.15$ points. Thus quadrupling the post-fork token budget beyond $4B$ adds only 0.19–0.28 points in total, while PSR at $16B$ remains 7.15–11.98%. Increasing the post-fork token budget helps at smaller values, but yields little further improvement in PSR within the tested range.

5 Implications and Limitations

The family results in Figure 4(a) identify tool/session changes and stale evidence as shared failure points. The trajectory analysis in Section 4.5 shows how agents can pass original task tests while removing valid collaborator work or relying on outdated evidence. These failures motivate checking preservation and evidence validity alongside task completion.

The evaluation covers 120 designed scenarios, some sharing task roots, with one update per run. Diagnostic controls cover five systems. Pairing fixes the recorded history and workspace state, while model randomness remains uncontrolled. Intervention position varies with execution stage, accumulated context, and remaining work. The post-fork token budget is fixed. Freshness checks verify required observations. Task and preservation checks assess the resulting state (Appendix D). Evaluating repeated changes in natural software-development workflows is a next step.

6 Conclusion

Existing repository-level coding benchmarks largely assume fixed tasks and environments, leaving agents' ability to handle mid-execution changes insufficiently tested. We introduced SWE-Shift to address this gap through task-relevant, recoverable interventions and matched Clean and Shifted continuations from the same saved execution state. Its success criteria jointly assess task completion, preservation of valid work, recovery, and evidence validity. Evaluation across 17 coding-agent systems shows that success under unchanged conditions often does not survive an update. Family-level and trajectory analyses further reveal recurring failures in state recovery, preservation, and evidence use that original task tests can miss. SWE-Shift provides a common testbed for measuring robustness to change, diagnosing these failures, and assessing whether new methods improve agents' ability to continue working reliably in evolving software environments.

AI use statement

OpenAI Codex assisted with manuscript organization, language editing, LaTeX formatting, writing code for figures and tables, reference verification, and benchmark data selection and construction.

For the benchmark experiments, we used Codex app for GPT-5.6 Sol and GPT-6 Astra, and Claude Code app for Claude Opus 5 and Claude Fable 5.1, all at `xhigh` reasoning. The open-weight coding-agent systems were served locally on four compute nodes with eight NVIDIA B300 GPUs each (32 GPUs total).

References

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1. Official provider announcement, 2026a. URL https://www.anthropic.com/claude-fable-and-mythos-5-1. Accessed September 21, 2026.

Anthropic. Introducing Claude Opus 5. Official provider announcement, 2026b. URL https://www.anthropic.com/news/claude-opus-5. Accessed September 21, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, et al. Measuring what matters: Construct validity in large language model benchmarks, 2025. URL https://arxiv.org/abs/2511.04703. NeurIPS 2025 Datasets and Benchmarks Track.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Published August 13, 2024; accessed September 26, 2026.

DeepSeek AI. DeepSeek-V4-Flash-0731 Model Card. Official provider model card, 2026a. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-0731/blob/7872f01b1d1fe23eabc4c98b48bffcef5a386062/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision 7872f01b1d1fe23eabc4c98b48bffcef5a386062.

DeepSeek AI. DeepSeek-V4-Pro Model Card. Official provider model card, 2026b. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/b5968e9190ef611bbf34a7229255be88a0e937c1/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision b5968e9190ef611bbf34a7229255be88a0e937c1.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments, 2026. URL https://arxiv.org/abs/2602.11964.

Shashwat Goel, Nikhil Chandak, Arvindh Arun, Ameya Prabhu, Steffen Staab, Moritz Hardt, Maksym Andriushchenko, and Jonas Geiping. FutureSim: Replaying world events to evaluate adaptive agents, 2026. URL https://arxiv.org/abs/2605.15188.

Xuehang Guo, Xingyao Wang, Yangyi Chen, Sha Li, Chi Han, Manling Li, and Heng Ji. SyncMind: Measuring agent out-of-sync recovery in collaborative software engineering. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20993–21066. PMLR, 2025. URL https://proceedings.mlr.press/v267/guo25l.html.

InternScience. Agents-A1 Model Card. Official provider model card, 2026. URL https://huggingface.co/InternScience/Agents-A1/blob/addff08f1653ee72765c5cf458fe84556bb34f8e/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision addff08f1653ee72765c5cf458fe84556bb34f8e.

Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. ACM, 2021. doi: 10.1145/3442188.3445901. URL https://www.microsoft.com/en-us/research/publication/measurement-and-fairness/.

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

Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors (ed.), *Belief Revision*, pp. 183–203. Cambridge University Press, 1992. doi: 10.1017/CBO9780511526664.007. URL https://doi.org/10.1017/CBO9780511526664.007.

Kabir Khandpur, Kilian Lieret, Carlos E. Jimenez, Ofir Press, and John Yang. SWE-bench Multilingual. SWE-bench dataset announcement, 2025. URL https://www.swebench.com/multilingual.html. Accessed September 26, 2026.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

MiniMax. MiniMax-M2.7 Model Card. Official provider model card, 2026a. URL https://huggingface.co/MiniMaxAI/MiniMax-M2.7. Official provider model card; accessed September 16, 2026.

MiniMax. MiniMax-M3-MXFP8 Model Card. Official provider model card, 2026b. URL https://huggingface.co/MiniMaxAI/MiniMax-M3-MXFP8/blob/c5454eb03678d8710e54a4e0fc681b9f3b4a3dba/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision c5454eb03678d8710e54a4e0fc681b9f3b4a3dba.

Mistral AI. Devstral-2-123B-Instruct-2512 Model Card. Official provider model card, 2025. URL https://huggingface.co/mistralai/Devstral-2-123B-Instruct-2512/blob/1613bf01adb5e1c6fdc196b46e6b173eae75eb4a/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision 1613bf01adb5e1c6fdc196b46e6b173eae75eb4a.

Andrey Mokhov, Neil Mitchell, and Simon Peyton Jones. Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):79:1–79:29, 2018. doi: 10.1145/3236774. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2018/03/build-systems-final.pdf.

Moonshot AI. Kimi-K2.6 Model Card. Official provider model card, 2026a. URL https://huggingface.co/moonshotai/Kimi-K2.6. Official provider model card; accessed September 16, 2026.

Moonshot AI. Kimi-K3 Model Card. Official provider model card, 2026b. URL https://huggingface.co/moonshotai/Kimi-K3/blob/f831ab66814297da540d832a5235f8e904f29d06/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision f831ab66814297da540d832a5235f8e904f29d06.

OpenAI. GPT-5.6 System Card. Official provider model card, 2026a. URL https://deploymentsafety.openai.com/gpt-5-6. Official provider model card; accessed September 16, 2026.

OpenAI. GPT-6 Astra System Card. Official provider system card, 2026b. URL https://deploymentsafety.openai.com/gpt-6-astra. Accessed September 21, 2026.

Qwen. Qwen3-30B-A3B-Instruct-2507 Model Card. Official provider model card, 2025. URL https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507. Official provider model card; accessed September 16, 2026.

Qwen. Qwen3.8-27B Model Card. Official provider model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.8-27B/blob/1d4bf0f2ff6012fd82039f2fa52739d0dd7c60c0/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision 1d4bf0f2ff6012fd82039f2fa52739d0dd7c60c0.

Qwen. Qwen3-Coder-Next Model Card. Official provider model card, 2026b. URL https://huggingface.co/Qwen/Qwen3-Coder-Next. Official provider model card; accessed September 16, 2026.

Mohit Raghavendra, Anisha Gunjal, Aakash Sabharwal, and Yunzhong He. SWE-INTERACT: Reimagining SWE benchmarks as user-driven long-horizon coding sessions, 2026. URL https://arxiv.org/abs/2606.30573.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Yuqiao Tan, Jinxiang Meng, Fangyu Lei, Minzheng Wang, Shizhu He, Jun Zhao, and Kang Liu. SWE-Touch: Benchmarking coding agents when users touch the code, 2026. URL https://arxiv.org/abs/2608.02499.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents, 2024. URL https://arxiv.org/abs/2407.18901.

Yangfan Wu, Haozhe Wang, Huanyu Yang, Jianmin Ji, and Fangzhen Lin. SpecPath: Testing coding agents across contract-equivalent specification histories, 2026a. URL https://arxiv.org/abs/2608.09799.

Yifan Wu, Zhuokai Zhao, Songlin Li, Ho Hin Lee, Jiacheng Zhu, Shirley Wu, Tianhe Yu, Serena Li, Lizhu Zhang, Xiangjun Fan, and Shengzhi Li. SWE-Together: Evaluating coding agents in interactive user sessions, 2026b. URL https://arxiv.org/abs/2606.29957.

Xiaomi MiMo. MiMo-V2.5-Pro Model Card. Official provider model card, 2026. URL https://huggingface.co/XiaomiMiMo/MiMo-V2.5-Pro/blob/21d1ecfecd7bd70f31be25ca49d7edd21f003659/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision 21d1ecfecd7bd70f31be25ca49d7edd21f003659.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Z.ai. GLM-5.3-Flash Model Card. Official provider model card, 2026. URL https://huggingface.co/zai-org/GLM-5.3-Flash/blob/690b705278a3a58e538fcb37c2ca8b5f9511213c/README.md. Official provider model card; accessed September 16, 2026; evaluated checkpoint revision 690b705278a3a58e538fcb37c2ca8b5f9511213c.

Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Linhao Zhang, Shulin Xin, Lu Chen, Qi Liu, Xiaojian Zhong, Aoyan Li, Siyao Liu, Yongsheng Xiao, Liangqiang Chen, Yuyu Zhang, Jing Su, Tianyu Liu, Rui Long, Kai Shen, and Liang Xiang. Multi-SWE-bench: A multilingual benchmark for issue resolving, 2025. URL https://arxiv.org/abs/2504.02605.

APPENDIX CONTENTS

A Extended Method **14**

A.1 What changes, and what must survive? 14

A.2 What is held fixed at the fork? 14

A.3 Construction checks 14

A.4 How T10–T90 is constructed and executed 15

B Extended Experimental Setup **15**

B.1 Systems and evaluation conditions 15

B.2 Success requires meeting all task requirements 15

B.3 Aggregation 16

C Construction and Validity **16**

C.1 Base task sources 16

C.2 Requirements by change family 16

C.3 Validation checks 16

C.4 Early completion 16

D Executable Success Checks **17**

E Extended Related Work and Benchmark Comparison **18**

F Coding-Agent Systems and Reproducibility **20**

G 24-Case Qualitative Audit Set **21**

A Extended Method

A.1 What changes, and what must survive?

A scenario specifies a repository task, a task-relevant change, how the agent can discover that change, and checks for a correct continuation. The change may affect repository files, runtime or session state, external effects, requirements, or the evidence available to the agent. After the change, some requirements remain, some are added, and some may be withdrawn. A correct continuation must satisfy the current requirements while preserving work and effects that are still valid.

The suite covers ten scenarios in each of 12 change families (Figure 3). F01–F07 concern code, collaborators, runtime, sessions, and effects; F08–F10 separate disclosure, corrective feedback, and goal revision; F11–F12 separate stale from misbound evidence. The 120 scenarios comprise an initial 24-case set and a 96-case expansion, sharing 23 root snapshots and 39 family–archetype–root clusters. The initial 24 cases were constructed and audited with GPT-5.6 Sol in the Codex app at `xhigh` reasoning; their full trajectories support the qualitative analysis in Appendix G. Roots can occur across families, and equal family weights provide controlled coverage of the change families.

A.2 What is held fixed at the fork?

For a system and checkpoint, the runner executes one prefix and saves the workspace, controller state, available tools, and the agent's recorded history, system settings, and counters. Both continuations resume from this saved state. Clean continues without a relevant update; Shifted applies the scenario's update. Their subsequent observations and actions may differ.

Each scenario identifies who can authorize an update and which requirements remain in force. The agent receives the task, allowed command and state-operation identifiers, and visible state. Changes appear through delivered messages, workspace differences, or permitted state queries. The evaluator's actor and version bindings are held out.

The agent uses structured file actions, named commands, and declared state operations. Available commands vary by scenario; some expose only repository status and diff, without a test command or unrestricted shell. The evaluator runs separate held-out checks. The evaluated coding-agent systems and their settings appear in Table 8.

A.3 Construction checks

Each admitted scenario requires a valid, task-relevant update and at least one preservation-safe recovery using only the agent-accessible tools. Table 3 lists the checks. Two negative examples test the update's relevance: a blind continuation must fail task or preservation checks, while a stale repair must fail at least one applicable Shifted check, even if it passes the original task tests.

Construction also checks affected dependencies, how the agent can discover the update, recovery order, information exposure, and deterministic restoration. Appendices C and D detail the family requirements and scoring checks.

Table 3: **Scenario admission checks.** Each scenario must pass all checks before evaluation.

State or execution	Required result
Initial root	Original task checks fail.
Clean reference repair	All Clean success checks pass.
Update without repair	External state remains valid; the task still fails.
Unadapted continuation	Clean succeeds; Shifted task or preservation checks fail.
Repair based on the pre-update state	Execution is protocol-valid; Shifted strict success fails.
Valid recovery after the update	Current task and preservation checks both pass.
Alternative valid recovery	A behaviorally distinct solution also passes.
Targeted invalid repairs	Task, preservation, and prohibited-effect errors are rejected.
Replay and restore	Isolated forks and repeated restoration give consistent results.

A.4 How T10–T90 is constructed and executed

The scenario-construction procedure starts from the case's canonical Clean reference action sequence, independently of the evaluated models' Shifted outcomes. It identifies the file and adapter dependencies of the update and its preconditions. The eligible prefix ends before the first action whose correctness depends on the update, including a read, tool call, or mutation, or at the end of the nonterminal reference sequence if no such action occurs.

If the prefix has fewer than nine actions, the procedure splits its longest eligible file read into contiguous, non-overlapping, nonempty line ranges, breaking ties by source order. The segments cover the original range exactly. A prefix that cannot yield nine real actions is rejected; no empty reads or no-ops are added. For N eligible actions, the checkpoint rule is

$$k_j = \left\lceil \frac{jN}{10} \right\rceil, \qquad j = 1, \ldots, 9, \quad |\{k_1, \ldots, k_9\}| = 9. \tag{7}$$

The nine checkpoints cover early through late reference progress, from 10% to 90% of the eligible action sequence. These fractions define the intervention schedule; they do not require evaluated agents to reach the same semantic stage of task completion. Replay checks the update and recovery at every checkpoint.

During evaluation, k_j specifies the number of actual agent actions before the fork. The agent follows its own trajectory, so the same checkpoint may correspond to different stages across systems. Early termination or actions that make the update inapplicable are handled by the outcome protocol at the assigned checkpoint, without rescheduling.

B Extended Experimental Setup

B.1 Systems and evaluation conditions

An evaluated system comprises the model, wrapper, backend, parser, tools, decoding policy, and execution limits. Table 8 lists the 17 coding-agent systems and their provider model cards. Model identities and available checkpoint revisions are recorded in `data/model-cards.json`. Each system uses the same 120 scenarios and nine intervention positions.

B.2 Success requires meeting all task requirements

For system m, scenario i, and checkpoint t, let ρ^a_{mit} be the trajectory of continuation a, including its observations, actions, modeled external effects, and terminal state. Let $\mathcal{J}^a_{mit}$ index the applicable binary checks v^a_{ij}. Raw strict success is

$$Y^a_{mit} = \prod_{j \in \mathcal{J}^a_{mit}} v^a_{ij}(\rho^a_{mit}), \qquad a \in \{\text{Clean}, \text{Shifted}\}. \tag{8}$$

Clean requires task and preservation checks, absence of prohibited effects, valid disclosure, current goal and verifier versions, and compliance with the post-fork token budget. Shifted additionally requires the recovery and fresh-observation checks. Only applicable checks enter the conjunction. Table 6 specifies the evidence used by each check.

Let C_{mit} and S_{mit} indicate verified strict success in Clean and Shifted, respectively. Success requires a protocol-valid execution. Joint success after intervention and total paired credit are

$$R_{mit} = C_{mit} S_{mit}, \qquad K_{mit} = \max\{R_{mit}, P_{mit}\}. \tag{9}$$

Here P denotes verified completion before the scheduled intervention, under the procedure in Appendix C. It is mutually exclusive with R, so $K = R + P$.

Clean, Shifted, and PSR retain all assigned cells in their averages; cells without verified credit contribute zero. Outcome records distinguish model failure, interference, infrastructure issues, and benchmark faults. Repaired or retried cells use their valid successor results while retaining the original fault and retry labels.

B.3 AGGREGATION

We weight each family equally, then each cluster within that family, then each scenario within the cluster. Let I_c contain the scenarios in cluster c and $\mathcal{C}_f$ the clusters in family f. For a per-cell quantity q,

$$H_{mt}(q) = \frac{1}{12} \sum_{f} \frac{1}{|\mathcal{C}_f|} \sum_{c \in \mathcal{C}_f} \frac{1}{|I_c|} \sum_{i \in I_c} q_{mit}. \tag{10}$$

The primary metric, Paired Success Rate (PSR), averages paired credit over the nine intervention positions:

$$\mathrm{PSR}_m = \frac{1}{9} \sum_{t \in \{T10,\ldots,T90\}} H_{mt}(K). \tag{11}$$

The same weights produce Clean, Shifted, and joint-success rates. At each checkpoint, $\mathrm{CRR}_{mt} = H_{mt}(R)/H_{mt}(C)$ when the denominator is nonzero. The overall CRR divides time-averaged joint success by time-averaged Clean success, excluding early-completion credit.

C CONSTRUCTION AND VALIDITY

C.1 BASE TASK SOURCES

The base repository tasks use pinned task instances from SWE-bench Verified, Multi-SWE-bench, and SWE-bench Multilingual. The 120 change scenarios share 23 task-root snapshots from 17 repositories; a root can support scenarios in more than one change family. Table 4 reports the scenario composition recorded in `data/benchmark.json`.

Table 4: **Base task sources for SWE-Shift.** Number of SWE-Shift change scenarios built from tasks in each source benchmark.

Base task source	Change scenarios
SWE-bench Verified (Chowdhury et al., 2024)	60
Multi-SWE-bench (Zan et al., 2025)	35
SWE-bench Multilingual (Khandpur et al., 2025)	25
Total	120

C.2 REQUIREMENTS BY CHANGE FAMILY

The taxonomy groups interventions by the dependency they change and the recovery they require. It draws on prior dynamic-agent evaluations and incidents encountered during development. Table 5 lists the requirements for each family.

C.3 VALIDATION CHECKS

Construction checks cover update preconditions, isolation of the paired continuations, reference recovery, preservation, information exposure, and replay/restore. Each scenario is checked at T0 and at all nine intervention positions T10–T90. All 120 T0 cases and 1,080 intervention cells pass these checks.

Each admitted scenario has a successful reference recovery using only the agent-accessible tools declared by that scenario (Appendix A). Reference recovery uses the same action interface as evaluated agents, without privileged state-setting operations. The evaluator checks the final state and event history independently, as specified in Appendix D.

C.4 EARLY COMPLETION

Early-completion credit requires replaying the completed prefix and verifying its task, preservation, and freshness conditions before the scheduled intervention. These cases receive P credit in PSR; the

Table 5: **Recovery requirements for the 12 change families.** Each row pairs a task-relevant change with the behavior a successful continuation must preserve or restore.

Family	What changes	Required continuation
F01 User implementation	Valid code is added in the agent's edit region.	Preserve the user's behavior while completing the original task.
F02 User contract	A public API or data contract changes.	Meet the updated contract and retained compatibility checks.
F03 Direct collaborator	A collaborator changes the target code or a direct dependency.	Integrate the changes while preserving the collaborator's behavior.
F04 Indirect collaborator	A transitive dependency changes while the local target stays untouched.	Account for the dependency change and preserve the collaborator's behavior.
F05 Runtime and resources	Process, port, lock, cache, or resource state changes.	Restore a valid runtime state while preserving unrelated resources.
F06 Tool and session	Session, authorization, schema, or protocol changes invalidate earlier calls.	Use the current permitted interface and preserve unrelated session state.
F07 External effects	An operation's actual outcome differs from what its response suggests.	Check the effect ledger, preserve valid effects, and perform only authorized compensation.
F08 Disclosure	A previously hidden constraint of a fixed goal becomes visible.	Meet the disclosed requirements under the same goal and verifier.
F09 Corrective feedback	New evidence corrects an earlier belief or implementation.	Apply and retain the correction while completing the unchanged goal.
F10 Goal revision	An authorized update revises the goal and verifier.	Follow the revised goal and preserve requirements that remain in force.
F11 Stale evidence	A result refers to an earlier state.	Obtain evidence for the current state before accepting completion.
F12 Misbound evidence	Evidence points to another request, branch, or session, or to conflicting sources.	Check its provenance through an allowed query and verify the intended state.

future update is not applied to the completed terminal state. CRR uses the joint-success component R.

D Executable Success Checks

The evaluator applies the checks in equation 8 to the workspace, controller, and modeled adapters. It also reads the event history to check action order and external effects. Table 6 summarizes the required evidence.

Table 6: **Evidence required for strict success.** Every applicable check must pass in a protocol-valid execution.

Check	Required evidence
Task	Executable task checks pass in the terminal state.
Preservation	Specified behavior and state invariants remain valid.
Prohibited effects	State and event checks find no forbidden outcome.
Current authority	Goal and verifier versions match the scenario's current requirements.
Fresh observation	An allowed observation occurs after the update and before the first agent mutation.
Recovery order	Required recovery steps occur in the specified order.
Recovery completion	Terminal recovery checks pass, and a state or adapter observation follows the last mutation.
Exposure and budget	Information access respects the declared limits, and post-fork token use stays within budget.

Freshness and post-repair observations are checked separately from task correctness. A state or adapter observation can satisfy the observation requirement; executable task and preservation checks assess the resulting state.

E Extended Related Work and Benchmark Comparison

Repository repair and shared editing. SWE-bench establishes executable repository-level issue resolution (Jimenez et al., 2024); SWE-agent demonstrates that the agent–computer interface changes how effectively a model can navigate, edit, and test a repository (Yang et al., 2024). SWE-Touch is the closest intervention study: it injects validated task-conflicting Counter-Edits during execution, includes a helpful non-solving Co-Edit control, and studies repeated edits and relative schedules on longer tasks (Tan et al., 2026). SyncMind uses repository histories to construct caller/callee inconsistencies and evaluates autonomous and collaborative recovery (Guo et al., 2025). SWE-Shift forks continuations from the same recorded history and checks the requirements that remain in force after an authorized update. An update may need to be retained even when removing it would still pass the original issue's tests. Table 7 compares these designs at the level of the manipulated object, history control, and outcome.

Table 7: **Different controls answer different questions about change.** "Started?" refers to evaluated execution before the described update; N/A marks a fixed-snapshot baseline. Each row lists what the cited protocol controls and checks. "Not reported" means that the source leaves a separate check unspecified.

Benchmark	Changed object	When	Started?	History control	Preservation / harm checks	Outcome
SWE-bench (Jimenez et al., 2024)	Repository repair	Initial task snapshot	N/A	Fixed issue and base commit	Previously passing tests	Resolved issue
SWE-Touch (Tan et al., 2026)	Code + contextual user message	Region access; relative schedules	Yes	Matched task conditions; independent runs	Task regression tests; Co-Edit control	Resolve, retention, steps, tokens
SyncMind (Guo et al., 2025)	Caller or callee made out of sync	Before recovery	Recovery starts stale	Rollback along Git history	Test-output consistency	Recovery, localization, assistance, cost
SWE-INTERACT (Raghavendra et al., 2026)	Requirements and feedback	During user dialogue	Yes	Single- and multi-turn task comparison	Original task verifier	Resolve, goal progress, interaction cost
SWE-Together (Wu et al., 2026b)	Requirements and corrective feedback	Reactive user turns	Yes	Reconstructed session intent; repeated runs	Frozen task rubric; separate external-work check not reported	Judge correctness; correction turns
SpecPath (Wu et al., 2026a)	Specification history	Before implementation	No	Equivalent final contract; fresh identical repository	Fixed final-contract probes	Realization; conditional path violation
ToolSandbox (Lu et al., 2025)	Stateful tool execution and implicit dependencies	During interactive dialogue	Yes	Stateful execution context with per-turn snapshots	Milestones, Minefields, and guardrail similarities over trajectories	Intermediate and final milestone satisfaction
Gaia2 (Froger et al., 2026)	App state and external events	Asynchronous event schedule	Yes	Initial state + event dependency graph	Write-action, causal, and temporal checks	Scenario success; action validity
AppWorld (Trivedi et al., 2024)	App database via agent actions	During API execution	Agent-driven	Initialized world; task contrast sets	Expected and allowed database differences	Task and scenario completion
SWE-Shift	Workspace, authority, runtime, or evidence	Nine eligible checkpoints	Yes	Same serialized prefix; Clean/Shifted fork	Task, preserved behavior, effects, and trace obligations	PSR, joint success, conditional retention

Interaction and specification histories. SWE-INTERACT progressively reveals requirements through a workspace-aware user simulator and scores the submitted solution with the underlying task verifier (Raghavendra et al., 2026). SWE-Together reconstructs tasks from real user–agent sessions, combines agentic-judge correctness with corrective-turn diagnostics, and grounds the simulator in the original user's intent (Wu et al., 2026b). SpecPath makes a complementary control explicit: vary the history while holding the final contract, repository, verifier, and execution policy fixed. Implementation begins only after the complete history is visible (Wu et al., 2026a). Its specification-history result and representation of active obligations are direct precedents for distinguishing retained from withdrawn requirements. SWE-Shift instead changes an already-started execution and its external state. Comparing these settings separates clarifying requirements before action from continuing work after a dependency or authority has changed.

State, side effects, and time. AppWorld already makes collateral damage part of task evaluation: expected and permitted database changes constrain the difference between initial and final states while admitting multiple solutions (Trivedi et al., 2024). ToolSandbox checks ordered milestones and forbidden events across stateful tool trajectories (Lu et al., 2025). τ^2-Bench gives both user and agent control of the environment and supports state and interaction checks (Barres et al., 2025). Gaia2 advances this setting to independently evolving, asynchronous events with action-level verification (Froger et al., 2026); FutureSim replays real information chronologically for forecasting agents (Goel et al., 2026). These works show why successful action cannot be reduced to a plausible final answer. SWE-Shift specializes this concern to controlled repository continuations, where preserving another actor's valid change and respecting an irreversible effect can require different recovery operations.

Behavioral testing and measurement validity. CheckList organizes targeted behavioral tests beyond aggregate accuracy (Ribeiro et al., 2020). Measurement modeling separates a claimed construct from its observable operationalization (Jacobs & Wallach, 2021); recent work applies this distinction directly to LLM benchmarks (Bean et al., 2025). This motivates the separation in our scoring between task tests, preservation checks, and records of required observations. Freshness checks record allowed observations; task and preservation checks assess behavior. Positive and selective-negative construction checks validate these measures before agent evaluation.

Updating information and rebuilding dependencies. The distinction between a changing world and new information about a fixed world predates language-model agents (Katsuno & Mendelzon, 1992). It helps keep true state changes separate from fixed-goal disclosure in our taxonomy. Build Systems à la Carte separates scheduling from deciding which dependent computations require rebuilding (Mokhov et al., 2018). These connections motivate checking which observations and derived results must be refreshed after an update.

F CODING-AGENT SYSTEMS AND REPRODUCIBILITY

Table 8: **Evaluated coding-agent systems.** The year links each official model reference. rc1* denotes vLLM `0.28.1rc1.dev452+g144e79c81`; dashes mark backend limits not exposed to the runner. Context and Max output are per-call limits in tokens; the post-fork token budget is $B = 1{,}000{,}000$ cumulative input plus output tokens per continuation.

System and model card	Inference path	Decoding	Context	Max output
GPT-6 Astra (2026b)	Codex app	xhigh	–	–
Claude Fable 5.1 (2026a)	Claude Code app	xhigh	–	–
Claude Opus 5 (2026b)	Claude Code app	xhigh	–	–
GPT-5.6 Sol (2026a)	Codex app	xhigh	–	–
Kimi-K3 (2026b)	vLLM 0.27.1	T=0, p=1	131,072	4,096
DeepSeek-V4-Pro (2026b)	vLLM 0.27.1	T=0, p=1	131,072	4,096
DeepSeek-V4-Flash (2026a)	vLLM rc1*	T=0, p=1	131,072	4,096
Kimi-K2.6 (2026a)	SGLang 0.5.17	T=0, p=1	131,072	4,096
MiniMax-M3 (2026b)	vLLM rc1*	T=0, p=1	131,072	4,096
GLM-5.3-Flash (2026)	vLLM rc1*	T=0, p=1	131,072	4,096
MiniMax-M2.7 (2026a)	SGLang 0.5.17	T=0, p=1	131,072	4,096
Devstral-2-123B-Instruct (2025)	vLLM rc1*	T=0, p=1	131,072	4,096
Agents-A1 (2026)	vLLM 0.28.0	T=0, p=1	131,072	4,096
Qwen3.8-27B (2026a)	vLLM 0.27.1	T=0, p=1	131,072	4,096
MiMo-V2.5-Pro (2026)	vLLM 0.27.1	T=0, p=1	131,072	4,096
Qwen3-30B-A3B-Instruct (2025)	SGLang 0.5.17	T=0, p=1	131,072	4,096
Qwen3-Coder-Next (2026b)	SGLang 0.5.17	T=0, p=1	131,072	4,096

Table 8 lists the 17 coding-agent systems. Model identifiers and available checkpoint revisions are provided in the accompanying system metadata file, `data/model-cards.json`.

The baseline post-fork token budget is $B = 1{,}000{,}000$ tokens per continuation, counted separately for Clean and Shifted as cumulative input plus output tokens across all model calls after the fork. The 4,096-token Max output for locally served systems in Table 8 limits a single model response, whereas the post-fork token budget covers all input and output tokens after the fork.

The reported tables and plots use `data/results.json`; the state-handling and post-fork token budget controls are in `data/controls.json`. The compact construction counts are in `data/benchmark.json`.

G 24-CASE QUALITATIVE AUDIT SET

We analyze the initial 24-case set, which contains two cases per family and complete trajectory and evaluator records. Each case occupies one page, pairing the task, authorized change, and requirements that apply after the change with selected actions from the recorded GPT-5.6 Sol trajectory (Codex app, `xhigh`) and independent evaluator checks. Step labels refer to the recorded trajectory; shaded monospace blocks show edit or tool-response excerpts, with selected response fields explicitly marked. The four-part result row reports Shifted task, recovery, preservation, and freshness; additional failed checks appear below it. Clean and Shifted denote success only when all applicable checks pass. Each card presents agent actions, self-reported test results, and independent evaluator findings separately.

No.	Family	Case / repository task	Clean	Shifted	Page
1	F01	F01-A07-SWE-201 / astropy/astropy	pass	pass	22
2	F01	F01-A07-REAL-202 / apache/druid	pass	pass	23
3	F02	F02-A08-REAL-201 / tokio-rs/bytes	fail	fail	24
4	F02	F02-A08-SWE-202 / sphinx-doc/sphinx	fail	fail	25
5	F03	F03-A07-REAL-201 / apache/druid	pass	pass	26
6	F03	F03-A07-REAL-202 / caddyserver/caddy	fail	fail	27
7	F04	F04-A06-SWE-201 / django/django	pass	fail	28
8	F04	F04-A06-REAL-202 / grpc/grpc-go	pass	pass	29
9	F05	F05-A03-REAL-201 / expressjs/express	fail	fail	30
10	F05	F05-A03-REAL-202 / fluent/fluentd	pass	fail	31
11	F06	F06-A09-REAL-201 / caddyserver/caddy	fail	fail	32
12	F06	F06-A09-SWE-202 / pylint-dev/pylint	fail	fail	33
13	F07	F07-A10-SWE-201 / astropy/astropy	pass	fail	34
14	F07	F07-A10-REAL-202 / tokio-rs/bytes	fail	fail	35
15	F08	F08-A10-SWE-201 / django/django	pass	fail	36
16	F08	F08-A10-SWE-202 / sympy/sympy	fail	fail	37
17	F09	F09-A07-SWE-201 / scikit-learn/scikit-learn	pass	fail	38
18	F09	F09-A07-REAL-202 / fluent/fluentd	pass	fail	39
19	F10	F10-A08-SWE-201 / pylint-dev/pylint	fail	fail	40
20	F10	F10-A08-SWE-202 / sphinx-doc/sphinx	fail	fail	41
21	F11	F11-A07-REAL-201 / grpc/grpc-go	fail	fail	42
22	F11	F11-A07-SWE-202 / scikit-learn/scikit-learn	fail	fail	43
23	F12	F12-A10-SWE-201 / sympy/sympy	fail	fail	44
24	F12	F12-A10-REAL-202 / expressjs/express	pass	fail	45

01 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Astropy: repair inside a surviving critical section

SSB-F01-A07-SWE-201
astropy/astropy

User implementation

STRICT SUCCESS

Clean ✓ PASS Shifted ✓ PASS

01 TASK

original request

Make property docstrings inherit through the InheritDocstrings metaclass.

02 AUTHORIZED SHIFT

A maintainer adds a lock-protected helper around docstring inheritance.

03 REQUIRED CONTINUATION

The old function-only eligibility test still needs repair; the new critical section and helper added by the maintainer must survive.

Trigger. The solver has located and read the existing InheritDocstrings member-selection loop and has formed a base-source repair plan, but has not edited or verified a repair.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Inspect repository status and who made the changes.

s9 Re-read the updated InheritDocstrings implementation.

s11 Widen the helper eligibility test to include properties.

Patch excerpt at s11 | `astropy/utils/misc.py`

```
- eligible = inspect.isfunction(val)
+ eligible = inspect.isfunction(val) or isinstance(val, property)
```

05 EVALUATION

independent checks

The task and all four preservation checks pass, including the complete critical section and the records showing who made the changes.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	✓ PASS	✓ PASS

02 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Druid: extend a guarded operation registry

SSB-F01-A07-REAL-202
apache/druid

User implementation

STRICT SUCCESS

Clean ✓ PASS Shifted ✓ PASS

01 TASK

original request

Add pow to arithmetic post-aggregation, including squares, cubes, and square roots.

02 AUTHORIZED SHIFT

A maintainer replaces unguarded lookup with a read/write-locked operation registry.

03 REQUIRED CONTINUATION

Adding an enum entry alone is insufficient: pow must be added to the guarded registry while retaining its locking behavior.

Trigger. The solver has inspected the Ops enum, cache-order switch, and lookup registry without editing or finishing.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read the changed operation registry and locking code.
- **s11** Add POW with Math.pow and registerBuiltIn(POW).
- **s12** Classify POW as order-sensitive in the cache-key switch.

Edit excerpt at s11 | `.../ArithmeticPostAggregator.java`

```
POW("pow") {
  @Override
  public double compute(double lhs, double rhs)
  {
    return Math.pow(lhs, rhs);
  }
};
```

05 EVALUATION

independent checks

Task, lookup read/write locks, and guarded pow registration pass.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	✓ PASS	✓ PASS

03 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Bytes: legacy repair meets a framed wire contract

SSB-F02-A08-REAL-201
tokio-rs/bytes

User contract

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Correct signed get_int/get_int_le decoding for widths below eight bytes.

02 AUTHORIZED SHIFT

A user introduces a V2 framed integer API with specified bytes, validation, and cursor behavior.

03 REQUIRED CONTINUATION

Legacy sign extension must be repaired while preserving the new reader/writer API requirements.

Trigger. The solver has inspected the variable-width signed getter and its macro-based decoding dependency and has not edited or finished.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s7 Inspect the shared-workspace diff before editing.

s16 Repair big-endian sign extension; repair little-endian at s17.

s18 Simplify the V2 decoder around the corrected signed payload.

Edit excerpt at s16 | `src/buf/buf_impl.rs`

```
let value = self.get_uint(nbytes);
if nbytes == 0 || nbytes == 8 {
    value as i64
} else {
    let shift = (8 - nbytes) * 8;
    (value << shift) as i64 >> shift
}
```

05 EVALUATION

independent checks

The original task and V2 recovery check pass, but the combined check for preserving the current API requirements and workspace fails.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

Evaluator finding: `private-evaluator test mutation has an unexpected digest`

04 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Sphinx: a compatibility boundary is bypassed

SSB-F02-A08-SWE-202
sphinx-doc/sphinx

User contract

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Include inherited data-member docstrings when autodoc searches base-class namespaces.

02 AUTHORIZED SHIFT

A user adds a versioned attribute-document snapshot API and an active discovery hook.

03 REQUIRED CONTINUATION

The repair must use the supported snapshot API rather than assume every analyzer exposes its earlier attributes.

Trigger. The solver has localized get_class_members and its ClassDocumenter caller but has not edited or run the verifier.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s10 Read the new attribute-document snapshot API.

s33 Traverse base-class namespaces, using analyzer.modname directly.

s38 Run the available tests after adding inherited-member coverage.

Edit excerpt at s33 | `sphinx/ext/autodoc/importer.py`

```
for cls in getmro(subject):
    if safe_getattr(cls, '__module__', None) == analyzer.modname:
        namespace = safe_getattr(cls, '__qualname__', None)
        if namespace:
            namespaces.append((namespace, cls))
```

05 EVALUATION

independent checks

The original task passes, but the API compatibility check reports that BoundaryAnalyzer has no modname attribute.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

Evaluator finding: `AttributeError: 'BoundaryAnalyzer' object has no attribute 'modname'`

05 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Druid: use the collaborator's operation abstraction

SSB-F03-A07-REAL-201
apache/druid

Direct collaborator

STRICT SUCCESS

Clean ✓ PASS Shifted ✓ PASS

01 TASK

original request

Add the pow arithmetic post-aggregation operation.

02 AUTHORIZED SHIFT

A collaborator refactors operation names and cache-order policy into OperationSpec.

03 REQUIRED CONTINUATION

The new arithmetic operation must use the shared metadata abstraction and keep order-sensitive cache behavior.

Trigger. The solver has inspected cache-order classification, Ops constructors, and quotient implementation without editing or finishing.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Read the collaborator's OperationSpec refactor.

s11 Define POW through OperationSpec with order sensitivity enabled.

s13 Run the available tests after adding pow coverage.

Edit excerpt at s11 | `.../ArithmeticPostAggregator.java`

```
POW(new OperationSpec("pow", true)) {
  @Override
  public double compute(double lhs, double rhs)
  {
    return Math.pow(lhs, rhs);
  }
};
```

05 EVALUATION

independent checks

Pow behavior, retained OperationSpec, and collaborator cache-order behavior all pass.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	✓ PASS	✓ PASS

06 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Caddy: changing defaults does not fix cookie behavior

SSB-F03-A07-REAL-202
caddyserver/caddy

Direct collaborator

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Set sticky cookies to Secure and SameSite=None by default.

02 AUTHORIZED SHIFT

A collaborator introduces sticky-cookie option and construction helpers.

03 REQUIRED CONTINUATION

Security defaults must take effect through the current cookie API while preserving the collaborator's helpers.

Trigger. The solver has localized the direct sticky-cookie construction and its request inputs without editing or finishing.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Inspect the collaborator's current diff.

s11 Set the helper defaults to Secure and SameSite=None.

s13 Add cookie assertions, then inspect the diff at s14.

Edit excerpt at s11 | `modules/caddyhttp/reverseproxy/selectionpolicies.go`

```
return stickyCookieOptions{
  secure:   true,
  sameSite: http.SameSiteNoneMode,
}
```

05 EVALUATION

independent checks

Collaborator registration remains, but the cookie API, task, and final behavior checks fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

07 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Django: tests pass after the policy API is removed

SSB-F04-A06-SWE-201
django/django

Indirect collaborator

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Reduce matching AddIndex/RemoveIndex migration operations.

02 AUTHORIZED SHIFT

A collaborator adds index_reduction_enabled and can_reduce_index_with as a shared policy API.

03 REQUIRED CONTINUATION

Index cancellation must honor the policy hook, which also governs other valid uses of the operation.

Trigger. The solver has localized AddIndex and the obvious Add/Remove cancellation point but has not edited or run the optimizer verifier.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s29** Initially call can_reduce_index_with in the reduction path.
- **s33** Remove the shared policy flag and method.
- **s34** Replace the policy call with direct model/name comparisons.

Edit excerpt at s34 | `django/db/migrations/operations/models.py`

```
if (
    isinstance(operation, RemoveIndex)
    and self.model_name_lower == operation.model_name_lower
    and self.index.name == operation.name
):
```

05 EVALUATION

independent checks

All 39 original task tests pass; preservation reports a missing can_reduce_index_with API and out-of-scope edits.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

Evaluator finding: `present=False`

08 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

gRPC: repair DNS failure through a surviving policy

SSB-F04-A06-REAL-202 Indirect collaborator
grpc/grpc-go

STRICT SUCCESS Clean ✓ PASS Shifted ✓ PASS

01 TASK

original request

Keep the last usable addresses when a temporary DNS failure occurs.

02 AUTHORIZED SHIFT

A collaborator adds policy-controlled lookup-error reporting.

03 REQUIRED CONTINUATION

The resolver must stop publishing an empty address list on errors while honoring the external reporting controls.

Trigger. The solver has localized dnsResolver.watcher and lookupHost as the DNS failure path but has not edited or run the verifier.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Inspect the resolver-maintenance diff.

s18 Report lookup errors through the surviving policy helper.

s19 Propagate errors from lookupHost instead of emitting empty state.

Edit excerpt at s18 | `internal/resolver/dns/dns_resolver.go`

```
state, err := d.lookup()
if err != nil {
  d.reportLookupError(err)
} else {
  d.cc.UpdateState(*state)
}
```

05 EVALUATION

independent checks

The task, temporary-DNS-failure recovery, policy API, and reporting behavior all pass.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	✓ PASS	✓ PASS

09 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Express: a replacement worker changes URL handling

SSB-F05-A03-REAL-201
expressjs/express

Runtime and resources

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Restore support for URL objects passed to res.location/res.redirect.

02 AUTHORIZED SHIFT

The runtime replaces the HTTP worker and installs a replacement URL-transport component.

03 REQUIRED CONTINUATION

URL normalization must work in the active worker while preserving the replacement transport and redirect behavior.

Trigger. The solver has inspected res.location and formed a clean-process URL-normalization plan, but has not edited or verified it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read res.location in the successor worker workspace.
- **s12** Replace the native-URL instanceof check with String(url).
- **s19** Run location tests after adding location and redirect coverage.

Patch excerpt at s12 | `lib/response.js`

```
- // The predecessor worker only normalizes native-realm URL
    values.
- var loc = url instanceof URL ? url.toString() : url;
+ var loc = String(url);
```

05 EVALUATION

independent checks

The active-worker task passes, but combined preservation fails and one prohibited-behavior event is recorded.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

Additional checks: prohibited-behavior count 1.

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

10 Fluentd: synchronization must retain worker-generation logic

SSB-F05-A03-REAL-202
fluent/fluentd

Runtime and resources

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Prevent ACK-handler shutdown races from raising unhandled exceptions.

02 AUTHORIZED SHIFT

The active worker is replaced and a worker-generation-aware ACK path is introduced.

03 REQUIRED CONTINUATION

Serializing socket collection must preserve checks tied to the replacement worker generation.

Trigger. The solver has inspected AckHandler.collect_response/read_ack_from_sock and formed a clean-worker race repair plan, but has not edited or verified it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s15 Inspect the worker-generation changes.

s17 Add a collection mutex; wrap collect_response at s18.

s22 Run available tests after adding concurrent-collection coverage.

Edit excerpt at s18 | `lib/fluent/plugin/out_forward/ack_handler.rb`

```
def collect_response(select_interval)
  @collect_mutex.synchronize do
    now = Fluent::Clock.now
    sockets = []
    results = []
```

05 EVALUATION

independent checks

Original task and replacement-worker activity pass, but the generation-aware ACK-path preservation check fails.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

Evaluator finding: `predecessor-generation ack was accepted`

11 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Caddy: the session changes around a cookie helper

SSB-F06-A09-REAL-201
caddyserver/caddy

Tool and session

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Set sticky cookies to Secure and SameSite=None.

02 AUTHORIZED SHIFT

A replacement cookie helper is installed and the tool proxy rebinds the session.

03 REQUIRED CONTINUATION

Edits and verification must apply to the replacement workspace and preserve its helper behavior.

Trigger. The solver has read the original cookie construction block and formed a direct-edit plan, but has not edited or tested it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read the current cookie-construction path.
- **s11** Set Secure and SameSite on newStickyCookie output.
- **s12** Add a helper test, then inspect the diff at s13.

Edit excerpt at s11 | `modules/caddyhttp/reverseproxy/selectionpolicies.go`

```
Path:     "/",
  Secure:   true,
  SameSite: http.SameSiteNoneMode,
}
```

05 EVALUATION

independent checks

The session and workspace bindings are correct; the helper and task checks fail, with one prohibited-behavior event.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

Additional checks: prohibited-behavior count 1.

12 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Pylint: an existing test result is not current acceptance

SSB-F06-A09-SWE-202
pylint-dev/pylint

Tool and session

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Disable duplicate-code checking when min-similarity-lines is zero.

02 AUTHORIZED SHIFT

The tool session changes, along with the interface used for similarity checking.

03 REQUIRED CONTINUATION

The repair must be accepted by the current task check rather than only a previously available test command.

Trigger. The solver has inspected Similar.run and formed a direct zero-threshold guard plan in the original workspace, but has not edited it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s6 Inspect the successor session state.

s19 Return an empty similarity result when min_lines is zero.

s22 Run the available test command; the final report cites 18 passes.

Edit excerpt at s19 | `pylint/checkers/similar.py`

```
if self.min_lines == 0:
    return []
```

05 EVALUATION

independent checks

The updated interface is preserved, but the original and current task checks fail. The terminal task run contains one failure.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	✓ PASS	✓ PASS

Evaluator finding: `adapter=tool_session_proxy matches=0`

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

13 Astropy: compensation succeeds, aggregate success does not

SSB-F07-A10-SWE-201
astropy/astropy

External effects

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Initially repair property-docstring inheritance. A later ledger entry reports cancellation.

02 AUTHORIZED SHIFT

Cancellation arrives after an external namespace-snapshot effect has committed, with a changed workspace still present.

03 REQUIRED CONTINUATION

Cancellation does not erase an already committed effect; compensation and the remaining workspace checks must be verified separately.

Trigger. The solver has localized the function-only member filter and formed a direct descriptor repair, while an external namespace-snapshot commit is still in flight.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read cancelled task authority and an already committed effect.
- **s8** Invoke ledger.compensate for the committed side effect.
- **s9** Re-read the ledger: the same commit is now compensated.

Selected response fields at s9 | `ledger.task-status`

```
{
  "authority.status": "cancelled",
  "side_effect.commit_id": "commit-000001",
  "side_effect.status": "compensated"
}
```

05 EVALUATION

independent checks

Cancellation authority, compensation, and commit linkage pass. The frozen task and workspace-preservation checks fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

14 Bytes: the compensated ledger does not undo the workspace

SSB-F07-A10-REAL-202
tokio-rs/bytes

External effects

STRICT SUCCESS — Clean × FAIL — Shifted × FAIL

01 TASK

original request

Initially correct short signed getters. A later ledger entry reports cancellation.

02 AUTHORIZED SHIFT

A signed-getter-related effect commits before cancellation, alongside a changed workspace.

03 REQUIRED CONTINUATION

The external ledger effect and the local code state require separate accounting after cancellation.

Trigger. The solver has localized both short signed getters and formed a helper-insertion plan while an external helper commit is still in flight.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read cancellation and the committed signed-getter effect.
- **s8** Invoke ledger.compensate.
- **s9** Confirm compensation; finish at s10 without a code edit.

Selected response fields at s9 | `ledger.task-status`

```
{
  "authority.status": "cancelled",
  "side_effect.commit_id": "commit-000001",
  "side_effect.status": "compensated"
}
```

05 EVALUATION

independent checks

Authority, exact compensation, and linkage pass; signed-getter task behavior and workspace preservation fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

Evaluator finding: `cancelled collaborator delta or another out-of-scope edit remains outside the two signed-getter bodies`

15 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Django: disclosure refines an unchanged goal

SSB-F08-A10-SWE-201 Disclosure
django/django

STRICT SUCCESS Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Reduce matching AddIndex/RemoveIndex migration operations.

02 AUTHORIZED SHIFT

An existing integration constraint becomes visible: exact index names, canonical model identity, and idempotence.

03 REQUIRED CONTINUATION

The goal is unchanged; the newly available constraint narrows which cancellation and model-folding behaviors are acceptable.

Trigger. The solver has localized AddIndex and formed the obvious name-only cancellation rule, but has not edited or run the optimizer verifier.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s30 Restore exact index-name equality in the reduction rule.

s31 Query requirements.current for the disclosed constraint.

s33 Extend CreateModel index folding beyond the cancellation rule.

Edit excerpt at s30 | `django/db/migrations/operations/models.py`

```
and self.index.name == operation.name
):
    return []
return super().reduce(operation, app_label)
```

05 EVALUATION

independent checks

Original and case task checks pass, and disclosure is recorded; preservation fails.

Task	Recovery	Preservation	Freshness
✓ PASS	× FAIL	× FAIL	✓ PASS

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

16 SymPy: a broader regex does not establish disclosed obligations

SSB-F08-A10-SWE-202
sympy/sympy

Disclosure

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Render numeric suffixes on Greek symbol names as Unicode subscripts.

02 AUTHORIZED SHIFT

A fixed-goal requirement about Unicode normalization and stable digit suffixes is revealed.

03 REQUIRED CONTINUATION

The repair must satisfy the current disclosed rule; widening character matching alone does not establish compatibility.

Trigger. The solver has localized the ASCII-only suffix regex and formed a Unicode-letter repair, but has not edited or tested it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s11 Read the original digit-suffix parsing expression.

s17 Replace the ASCII-only prefix with a Unicode-letter pattern.

s19 Add pretty-print assertions; finish without querying requirements.

Patch excerpt at s17 | `sympy/printing/conventions.py`

```
- _name_with_digits_p = re.compile(r'^([a-zA-Z]+)([0-9]+)$')
+ _name_with_digits_p = re.compile(r'^([^\W\d_]+)([0-9]+)$')
```

05 EVALUATION

independent checks

Task, recovery, observation-order freshness, and disclosure validity fail; preservation passes.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	✓ PASS	× FAIL

Additional checks: disclosure fail.

Evaluator finding: `adapter=requirement_graph matches=0`

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

17 Scikit-learn: a correct answer can exceed the allowed edit scope

SSB-F09-A07-SWE-201
scikit-learn/scikit-learn

Corrective feedback

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Include joblib in show_versions dependency reporting.

02 AUTHORIZED SHIFT

A second review requires joblib immediately after sklearn, with prior ordering and missing-module behavior retained.

03 REQUIRED CONTINUATION

The new ordering obligation must coexist with optional imports and the permitted repository scope.

Trigger. The solver has read _get_deps_info and formed a dependency-list edit plan, but has not yet edited or verified it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Read the second-round feedback.

s10 Insert joblib immediately after sklearn; acknowledge at s11.

s13 Also edit the test file, which later fails the write-scope check.

Edit excerpt at s10 | `sklearn/utils/_show_versions.py`

```
"sklearn",
"joblib",
"numpy",
```

05 EVALUATION

independent checks

Canonical joblib task behavior passes. Preservation reports an unexpected tracked diff scope including the added test edit.

Task	Recovery	Preservation	Freshness
✓ PASS	✓ PASS	× FAIL	✓ PASS

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

18 Fluentd: retain failure semantics while quieting shutdown noise

SSB-F09-A07-REAL-202
fluent/fluentd

Corrective feedback

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Handle closed ACK sockets during shutdown without crashing.

02 AUTHORIZED SHIFT

A second review requires debug-level handling of expected closed sockets while retaining earlier nil-safe retry behavior.

03 REQUIRED CONTINUATION

Reducing log severity must not erase failed-delivery semantics for missing ACK information.

Trigger. The solver has inspected the ACK select/read race and formed the upstream nil-safe IOError repair plan, but has not edited or tested it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read the new feedback; inspect feedback history at s12.
- **s17** Add IOError-specific debug logging, also at s18.
- **s31** Return nil when find(sock) has no ACK metadata.

Edit excerpt at s31 | `lib/fluent/plugin/out_forward/ack_handler.rb`

```
info = find(sock)
return nil unless info
```

05 EVALUATION

independent checks

The nil-safe race check passes; missing-ACK failure preservation and exclusive-debug task checks fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

Evaluator finding: `missing ACK metadata must remain failed, got nil`

19 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

Pylint: revised zero mode reaches beyond ingestion

SSB-F10-A08-SWE-201
pylint-dev/pylint

Goal revision

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Originally disable duplicate detection at zero; the revised goal demands a true ingestion no-op.

02 AUTHORIZED SHIFT

An authorized goal update forbids stream reads, decoding, and LineSet allocation at zero, retaining nonzero behavior.

03 REQUIRED CONTINUATION

The zero-mode requirements must hold across the relevant execution path, not just one early return.

Trigger. The solver has inspected Similar.append_stream and Similar.run and formed the original output-suppression plan, but has not edited or tested it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s9 Read append_stream in the similarity checker.

s10 Add an immediate return for min_lines == 0.

s15 Add an ingestion regression test; run available tests at s16.

Edit excerpt at s10 | `pylint/checkers/similar.py`

```
if self.min_lines == 0:
    return
if encoding is None:
```

05 EVALUATION

independent checks

The current goal is bound correctly, but the behavioral check reports ”zero mode computed similarities”; preservation also fails.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	✓ PASS

Evaluator finding: `zero mode computed similarities`

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

20 Sphinx: per-call reuse must retain nearest-class precedence

SSB-F10-A08-SWE-202
sphinx-doc/sphinx

Goal revision

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Support inherited instance-attribute documentation.

02 AUTHORIZED SHIFT

The revised request limits parsing to once per module per call and requires nearest-MRO precedence without a global cache.

03 REQUIRED CONTINUATION

Cross-class discovery, local reuse, and duplicate-attribute precedence must hold together.

Trigger. The solver has read get_class_members and its ClassDocumenter caller and formed the original MRO analyzer plan, but has not edited or tested it.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s8 Read the inherited-member discovery path.

s24 Cache analyzers per call and gather attributes across the MRO.

s26 Run the available test command before finishing.

Edit excerpt at s24 | `sphinx/ext/autodoc/importer.py`

```
analyzers = {subject.__module__: analyzer}
attr_docs = {}
for cls in getmro(subject):
    modname = cls.__module__
    if modname not in analyzers:
```

05 EVALUATION

independent checks

The original task, goal binding, and workspace scope pass; the revised inherited-attribute check fails with a missing shared entry.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	✓ PASS	✓ PASS

Evaluator finding: `KeyError: 'shared'`

21 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

gRPC: code repair leaves a stale evidence route

SSB-F11-A07-REAL-201 Stale evidence
grpc/grpc-go

STRICT SUCCESS Clean × FAIL Shifted × FAIL

01 TASK

original request

Avoid clearing usable resolver addresses during temporary DNS errors.

02 AUTHORIZED SHIFT

The evidence route switches to a stale artifact; executable world state is unchanged.

03 REQUIRED CONTINUATION

Evidence used to accept the repair must be current even when the repository itself has not changed.

Trigger. The solver has inspected dnsResolver.watcher and lookupHost and is about to decide whether the existing resolver behavior already passes.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s8** Read the current watcher and resolver error path.
- **s19** Propagate host lookup errors; call ReportError at s20.
- **s21** Run available tests; no route-restoration action is recorded.

Edit excerpt at s20 | `internal/resolver/dns/dns_resolver.go`

```
state, err := d.lookup()
if err != nil {
  d.cc.ReportError(err)
} else {
  d.cc.UpdateState(*state)
}
```

05 EVALUATION

independent checks

Route mode remains stale rather than current; task and workspace-scope checks also fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	× FAIL

Evaluator finding: `adapter=evidence_router actual='stale' expected='current'`

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

22 Scikit-learn: restoring the route is not refetching evidence

SSB-F11-A07-SWE-202
scikit-learn/scikit-learn

Stale evidence

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Include joblib in show_versions dependency reporting.

02 AUTHORIZED SHIFT

The task-evidence route becomes stale while the task and executable world remain fixed.

03 REQUIRED CONTINUATION

Selecting the current route must be followed by acquiring evidence from that route; editing code alone cannot supply that observation.

Trigger. The solver has localized _get_deps_info and is about to decide whether a code change is still needed.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s7 Restore the evidence route; the response reports mode=current.

s9 Add joblib to the reported dependency list.

s12 Run available tests; no evidence-router refetch is recorded.

Recorded response at s7 | `evidence.restore-current-route`

```
{
  "cache_epoch": 2,
  "mode": "current",
  "route_id": "task-evidence"
}
```

05 EVALUATION

independent checks

Task and current-route identity pass; current-evidence refetch, freshness, disclosure, and write-scope requirements fail.

Task	Recovery	Preservation	Freshness
✓ PASS	× FAIL	× FAIL	× FAIL

Additional checks: disclosure fail.

Evaluator finding: `adapter=evidence_router matches=0`

23 24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

SymPy: an unchanged task with cross-session evidence

SSB-F12-A10-SWE-201
sympy/sympy

Misbound evidence

STRICT SUCCESS

Clean × FAIL Shifted × FAIL

01 TASK

original request

Render Greek symbol suffixes as Unicode subscripts.

02 AUTHORIZED SHIFT

Asynchronous evidence binding switches to cross-session mode and the route is misbound.

03 REQUIRED CONTINUATION

An observation must belong to the current execution, even when the desired rendering and repository are unchanged.

Trigger. The solver has read split_super_sub and the trailing-digit regex and is about to use asynchronous verifier evidence to decide whether to edit.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

s9 Read the trailing-digit parser in conventions.py.

s15 Broaden the prefix regex; add a pretty-print assertion at s16.

s17 Run available tests; no binding-restoration action is recorded.

Patch excerpt at s15 | `sympy/printing/conventions.py`

```
- _name_with_digits_p = re.compile(r'^([a-zA-Z]+)([0-9]+)$')
+ _name_with_digits_p = re.compile(r'^([^\W\d_]+)([0-9]+)$')
```

05 EVALUATION

independent checks

Binding remains cross_session and route remains misbound; task and preservation fail.

Task	Recovery	Preservation	Freshness
× FAIL	× FAIL	× FAIL	× FAIL

Evaluator finding: `adapter=async_binding actual='cross_session' expected='strict'`

24-CASE QUALITATIVE AUDIT SET / CASE EVIDENCE

24 Express: task success with the wrong evidence binding

SSB-F12-A10-REAL-202
expressjs/express

Misbound evidence

STRICT SUCCESS

Clean ✓ PASS Shifted × FAIL

01 TASK

original request

Accept URL objects in location and redirect responses.

02 AUTHORIZED SHIFT

The asynchronous evidence adapter switches to cross-session binding; the executable world is unchanged.

03 REQUIRED CONTINUATION

Behavioral correctness must be supported by evidence bound to the current process and permitted workspace scope.

Trigger. The solver has inspected res.location, the lower-case scheme check, and the encoding path and is about to use process-scoped verifier evidence to decide whether URL objects already work.

04 OBSERVED TRAJECTORY

GPT-5.6 Sol (xhigh)

- **s7** Read res.location and its URL normalization path.
- **s13** Convert loc to a string before scheme checks and encoding.
- **s19** Run location tests; no binding correction is recorded.

Edit excerpt at s13 | `lib/response.js`

```
loc = String(loc);

var lowerLoc = loc.toLowerCase();
var encodedUrl = encodeUrl(loc);
```

05 EVALUATION

independent checks

Location behavior and original task pass, but binding remains cross_session instead of strict, and workspace scope fails.

Task	Recovery	Preservation	Freshness
✓ PASS	× FAIL	× FAIL	× FAIL

Evaluator finding: `adapter=async_binding actual='cross_session' expected='strict'`