WORKROVER: SCALING LONG-HORIZON COWORK AGENT TRAINING VIA RECURSIVE ENVIRONMENT EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling interactive environments has become increasingly important for training long-horizon agents through reinforcement learning. These environments often require further testing and refinement after initial generation, and correcting defects in a task may also require updating its work context and verifier. However, existing construction pipelines offer limited support for such iterative refinement while keeping these components aligned. In this paper, we introduce **WORKROVER**, a framework for training long-horizon cowork agents through recursive environment evolution. Its harness-driven synthesis enables construction agents to jointly build and validate semantically aligned work contexts, executable tasks, and evidence-grounded verifiers, adapting their actions to workspace state and execution feedback rather than following a fixed construction workflow. To support environment evolution, we introduce **Factorial Compatibility Replay (FCR)**, which uses rollout feedback to evaluate combinations of component revisions through dependency-closed replay and selects the smallest validated repair among tested candidates while preserving task requirements and evaluation constraints. To provide finer-grained training signals, we propose **Verifier-to-Turn Credit Assignment (VTCA)**, which augments trajectory-level GRPO advantages with signals from deterministic workspace-progress checks and assigns a bounded share of positive-progress credit to prerequisite turns using recorded tool-output and artifact dependencies. Using WORKROVER, we post-train Qwen3.5-35B-A3B to obtain WORKROVER-35B-A3B and evaluate it on several cowork-agent benchmarks. Its reported scores exceed those of Qwen3.5-35B-A3B by 26.27 percentage points on JobBench, 29.32 percentage points on OfficeQA Pro, and 413.25 Elo points on GDPval v2. These results suggest that WORKROVER offers a promising foundation for scalable training of long-horizon cowork agents.

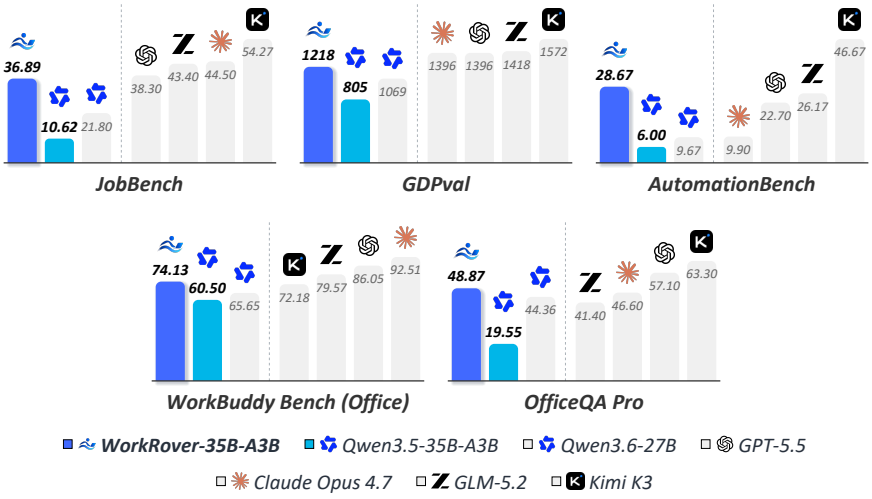

Figure 1: **Cross-benchmark performance.** Results on five cowork-agent benchmarks.

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.

1 INTRODUCTION

Cowork agents powered by large language models (LLMs) perform professional tasks across files, tools, and applications. These tasks unfold over long sequences of dependent actions, requiring agents to locate relevant materials, track workspace changes, and maintain consistency across related artifacts (Niu et al., 2026; Tang et al., 2026). Training these agents through reinforcement learning (RL) depends on environments that support diverse, executable workflows and provide reliable feedback on task outcomes (Hao et al., 2026; Cheng et al., 2026a). Recent work has explored automated environment and task synthesis to expand the range of workspaces and tasks available for agent training (Song et al., 2026b; Bai et al., 2026).

However, expanding the number of synthesized environments does not by itself ensure reliable training feedback. In a cowork environment, the work context must support the task's requirements, and the verifier must assess the requested outcomes while accepting valid alternative solutions. Agent rollouts can expose missing dependencies, inconsistent task specifications, or verification errors that remain after initial construction (Zhong et al., 2026b). Such defects can make valid behavior appear unsuccessful or allow incorrect results to be rewarded. Repairing these defects often requires coordinated revisions within a task environment: a correction to the work context or task specification may also require updating the verifier to keep execution and evaluation aligned. Existing construction pipelines offer limited support for maintaining this alignment throughout repeated revisions. This motivates *recursive environment evolution*: repeatedly using execution feedback to identify defects, coordinate component revisions, and revalidate the revised environment while preserving the original task requirements and evaluation constraints. Reliable terminal evaluation also leaves a credit-assignment challenge: useful steps can occur in failed trajectories and redundant actions in successful ones, while earlier information-gathering turns may only contribute through later workspace changes (Tao et al., 2026).

To this end, we introduce **WORKROVER**, a framework for long-horizon cowork agent training that integrates harness-driven environment synthesis, recursive environment evolution, and verifier-grounded policy learning (Figure 2(b)). Its harness tracks component dependencies and enables construction agents to jointly build and validate semantically aligned work contexts, executable tasks, and evidence-grounded verifiers. These agents adapt their construction actions to workspace state and execution feedback rather than following a fixed workflow. Independent reference executions from a clean initial workspace establish task feasibility, while verifier validation tests incorrect outcomes and valid alternatives against instruction-grounded requirements rather than a prescribed solution path. For environment evolution, we introduce *Factorial Compatibility Replay* (FCR), which evaluates combinations of fixed repair proposals across the work context, task, harness, and verifier. Each combination is rebuilt with its dependency-induced updates and tested under matched rollout conditions. FCR selects the smallest validated repair among the tested combinations, subject to preserving task requirements and evaluation constraints and passing regression checks on affected tasks. Fresh rollouts confirm selected repairs and provide evidence for subsequent evolution. For policy learning, we introduce *Verifier-to-Turn Credit Assignment* (VTCA) to supplement trajectory-level GRPO advantages with finer-grained feedback. Frozen deterministic checks measure progress and setbacks in recorded workspace states, while tool-output and artifact dependencies identify prerequisite turns eligible for a bounded share of positive-progress credit. This provides turn-level learning signals without intermediate judge calls.

We post-train Qwen3.5-35B-A3B with WORKROVER and evaluate the resulting model on five external benchmarks spanning occupational tasks, professional deliverables, cross-application workflows, and document-grounded reasoning. WORKROVER-35B-A3B exceeds the reported backbone scores on all five benchmarks, with gains of 26.27 percentage points on JobBench, 29.32 percentage points on OfficeQA Pro, and 413.25 Elo points on GDPval v2. Controlled experiments further examine the contributions of environment evolution and verifier-grounded learning. Under matched repair budget caps, FCR improves task-package synthesis pass rate by 6.0 percentage points over full regeneration. A separate maintenance evaluation shows that FCR also resolves naturally occurring defects more reliably while reducing unintended changes to task intent. Construction ablations also show that following file dependencies and matching files with compatible skills improve task yield and executability, beyond assembling workspaces through topical similarity alone. With the supervised fine-tuning (SFT) initialization, task and seed schedules, and rollout budget fixed, VTCA outperforms episode-only GRPO across all five benchmarks. These findings support improving both environment reliability and turn-level learning signals for long-horizon cowork agent training.

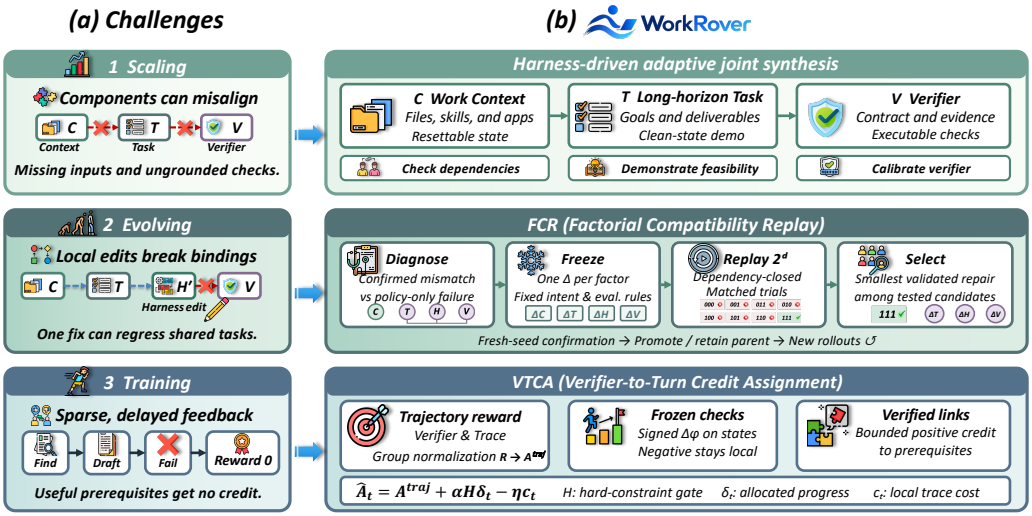

Figure 2: **Cowork agents: challenges and approaches.**

Our main contributions are:

1. **Harness-driven construction of semantically aligned environments.** We introduce a Harness-driven synthesis pipeline that jointly constructs *semantically aligned* work contexts, executable tasks, and evidence-grounded verifiers at scale. The Harness explicitly *tracks their dependencies* and validates *task feasibility and verifier consistency*.
2. **Recursive environment evolution.** We introduce *Factorial Compatibility Replay* (FCR) to support iterative environment repair under explicit component dependencies. FCR evaluates coordinated revisions, identifies *minimal validated repairs*, and propagates changes across affected components while *preserving task intent and evaluation constraints*.
3. **Training from verifier-grounded progress.** We introduce *Verifier-to-Turn Credit Assignment* (VTCA) to augment terminal-reward GRPO with *workspace-grounded advantage shaping*. Deterministic checks provide progress proxies, with bounded positive allocations to *prerequisite turns* through recorded tool and artifact dependencies, without intermediate Judge calls.

2 Related Work

Executable environments and verification. Prior work scales executable environments (Trivedi et al., 2024; Song et al., 2026b; Bai et al., 2026; Dong et al., 2026b) and validated task synthesis (Chen et al., 2025; Pan et al., 2026b; Hua et al., 2026b). Shared specifications and state help align tasks and verifiers (Ivanov & Rana, 2026; Shi et al., 2026), while adversarial audits test verifier reliability (Zhong et al., 2026b). WorkRover jointly constructs Work Contexts, Tasks, and Verifiers, tracking their dependencies under a versioned Harness.

Environment evolution and repair. Recent work studies task-bundle revision and execution-driven environment and harness repair (Wu et al., 2026b; Pandya et al., 2026; Lin et al., 2026a). Program repair also studies patch minimization (Weimer et al., 2009). FCR replays combinations of fixed repair proposals, including dependency-induced changes. It selects the smallest validated repair among those tested, preserving task requirements and evaluation constraints and checking affected tasks for regressions.

Long-horizon learning and progress attribution. Local credit assignment draws on state comparisons and planning hierarchies (Feng et al., 2025; Peng et al., 2026a), progress estimates, or evidence attribution (Tao et al., 2026; Zhang et al., 2026b; Xu et al., 2026a; Gandhi et al., 2026). VTCA shapes GRPO advantages (Shao et al., 2024) with frozen checks of workspace progress and setbacks. It assigns a bounded share of positive credit to prerequisite turns using recorded tool-output and artifact-version dependencies, without intermediate Judge calls. Checker coverage (Zhou et al., 2026a) and dependency quality limit these signals. See Appendix A.

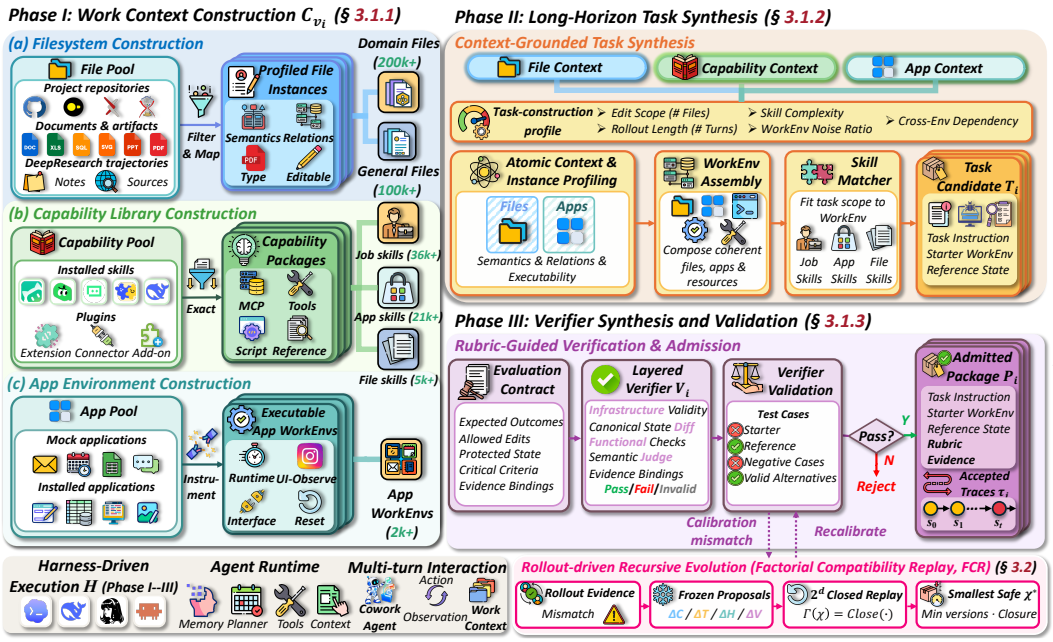

Figure 3: Environment construction and recursive evolution in WORKROVER. The Harness coordinates Work Context construction, Task synthesis, and Verifier synthesis and validation. FCR repairs compatibility mismatches under frozen safety gates; verified trajectories support policy learning.

3 METHOD

In this section, we present **WORKROVER** (Figure 3), a framework for training long-horizon cowork agents through recursive environment evolution. We first describe harness-driven synthesis of semantically aligned work contexts, tasks, and verifiers (Sec. 3.1). We then present **Factorial Compatibility Replay (FCR)** for rollout-driven recursive evolution (Sec. 3.2) and **Verifier-to-Turn Credit Assignment (VTCA)** (Sec. 3.3).

3.1 HARNESS-DRIVEN ENVIRONMENT SYNTHESIS

3.1.1 WORK CONTEXT CONSTRUCTION

We formulate environment construction for long-horizon cowork training as the joint synthesis of work contexts, tasks, and verifiers. Each environment must support task execution and provide reliable feedback on the result. We organize the training collection as N task packages:

$$\mathfrak{L} = \{\mathcal{P}_i = \langle C_i, T_i, V_i \rangle\}_{i=1}^{N}.$$

The Work Context C_i provides files, skills, and application environments, organized into File, Capability, and App Contexts. The task T_i specifies goals, deliverables, and constraints. The Verifier V_i checks whether the agent's work meets these requirements. The Harness H coordinates construction, tracks component dependencies, and manages execution.

Phase I builds a reusable Work Context $C = (C^{\text{file}}, C^{\text{cap}}, C^{\text{app}})$. The Skill Matcher uses job skill requirements to retrieve related files from graphs that record content similarity, cross-file references, and workflow dependencies. It then matches these materials with file and application skills, checking compatibility across their inputs, outputs, and interfaces. The Harness then prepares the workspace so that the selected skills can access the required files and operate the corresponding applications. See details in Appendices B.1 and C.2.

3.1.2 LONG-HORIZON TASK SYNTHESIS

Context profiling and WorkEnv assembly. In Phase II, we synthesize task T_i from the File, Capability, and App Contexts in C_i. The construction profile sets the target number of edited files and interaction turns, the skill combinations, the proportion of distractors, and dependencies across environments (Table 6; Appendix B.2). *Atomic Context & Instance Profiling* identifies what each file or app contains, how it relates to other resources, and which operations it supports. The Harness then assembles related files, apps, and capabilities into a WorkEnv under H.

Skill matching and task specification. Starting from job skill j, the Skill Matcher combines job, file, and app skills supported by the assembled WorkEnv. It links these skills into an internal workflow $\Pi_i = (s_{i,1}, \ldots, s_{i,m_i})$, where later stages depend on earlier outputs or state changes. The dependency graph D_{T_i} records the files, capabilities, and application states each stage needs. The Harness uses it to prepare the Starter WorkEnv x_i^0, including all dependencies and marking editable and protected state. Instructions state goals, materials, deliverables, and constraints without revealing paths, tool or skill names, or prescribed actions.

Reference execution and task candidates. A successful independent Demonstrator rollout from x_i^0 establishes feasibility and yields a Reference State $x_i^\star$. The candidate contains three items: Task Instruction, Starter WorkEnv, and Reference State. Changes between the two states identify candidate verification evidence; the workflow, Reference State, and trace cannot add requirements or exclude valid alternatives. We remove inputs to test whether each dependency is necessary. Independently diagnosed construction mismatches trigger candidate-stage FCR (Section 3.2). Candidates that pass construction checks enter Phase III for Verifier synthesis and validation.

3.1.3 Verifier Synthesis and Validation

Evaluation Contract. We derive criteria from task instructions and declared workspace conditions visible to the agent, fixing them before consulting the Reference State. The Contract records expected outcomes, allowed edits, protected state, and admissible evidence. The Starter WorkEnv and Reference State provide evidence for these checks without adding requirements or prescribing a solution.

Layered verification. The Verifier follows the four layers in Figure 3. *Infrastructure Validity* returns $invalid$ when infrastructure faults prevent evaluation. *Canonical State Diff* checks state changes against the Contract. *Functional Checks* verify artifact content and consistency, while the *Semantic Judge* handles criteria that executable checks cannot resolve. A valid rollout passes when its weighted outcome score meets the threshold and all mandatory checks pass; otherwise it fails.

Verifier validation and admission. We calibrate the Verifier to reject the Starter State and negative cases that violate passing conditions, while accepting the Reference State and independent valid alternatives. Independently confirmed mismatches may trigger FCR, followed by recalibration under the fixed Contract. We then freeze the Verifier for a separate, one-time admission test. Failure rejects the candidate without repair feedback. Full scoring and validation details are in Appendix B.3.

3.2 Recursive Environment Evolution

Factorial Compatibility Replay (FCR) repairs problems revealed during construction or later rollouts. Because the Work Context, Task, Harness, and Verifier depend on one another, FCR compares coordinated changes and selects the smallest validated repair while preserving task intent.

Diagnosis and proposals. Independent diagnostics distinguish component problems from policy failures; policy failures alone leave packages unchanged. The Harness records affected versions and dependencies, fixes task requirements and validation checks, then prepares and freezes one proposal per eligible component. The Evaluation Contract remains fixed once grounded. Agent-visible edits must preserve information boundaries and pass solver-panel checks against task simplification.

Harness screening and replay. Each proposal can be applied or omitted, so the candidate space grows exponentially. To limit replay cost, the Harness screens combinations before rebuilding packages or running rollouts. It rejects confirmed dependency, binding, or requirement conflicts; uncertain cases proceed to validation. Surviving combinations are rebuilt with all required changes to dependent components, including those without a selected direct edit. Compatible versions are replayed using the same policies, budgets, and seeds. Comparing proposals separately and together reveals which repairs need one another. When only the Verifier differs, alternatives score identical recorded evidence.

Repair selection. Independent checks must confirm that a repair resolves the problem, preserves required outcomes and information boundaries, and causes no regressions in affected packages. Resolution checks use neither policy success nor candidate-Verifier scores. Once a Verifier exists, candidates must also pass the fixed Contract checks and tests of known failures and valid alternatives. FCR selects the candidate with the fewest component versions changed, including changes induced by dependencies; ties favor fewer rebuilt dependencies, then lower cost.

Confirmation and update. Fresh rollouts with new seeds confirm the selected repair. Unadmitted candidates repeat validation from the earliest affected construction step and must pass initial admission; admitted packages activate all validated changes together. If selection or confirmation fails,

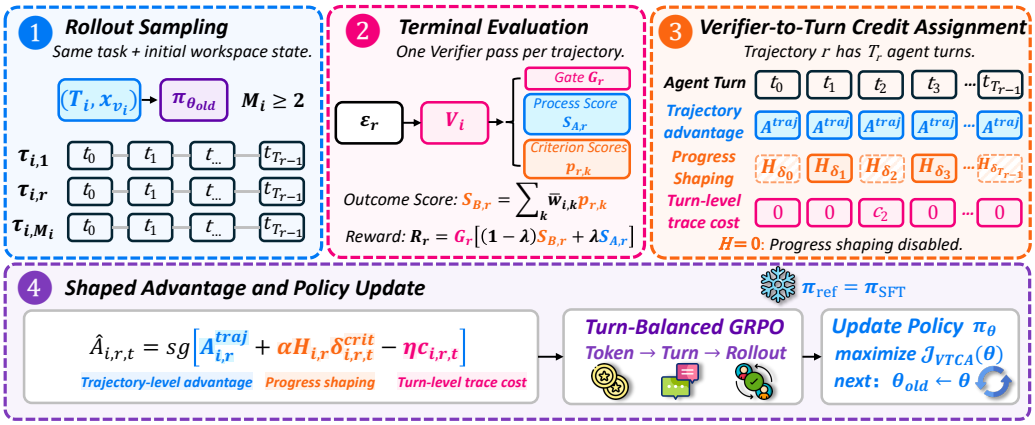

Figure 4: **Overview of VTCA.** Same-task rollouts yield terminal rewards and checked progress. Bounded prerequisite allocations and local trace costs shape turn-balanced GRPO advantages.

FCR discards an unadmitted candidate or retains the admitted parent. Subsequent rollouts guide the next repair cycle. See Appendices B.4 and B.5 for details.

3.3 Verifier-to-Turn Credit Assignment

Terminal rewards score an entire rollout but provide little guidance about which turns made progress. VTCA adds feedback from verified workspace changes and recorded dependencies (Figure 4).

Trajectory supervision. We initialize the policy with verified demonstrations using SFT, then sample RL rollouts for the same task and initial state. Among rollouts with valid evaluations, terminal rewards combine Verifier outcome scores and trace scores. Rewards are zero if any critical criterion or integrity, protected-state, or safety check fails. We standardize rewards within each group to obtain the trajectory advantage $A^{\text{traj}}_{i,r}$ for task i and rollout r.

Measuring progress. Frozen deterministic checks (Tao et al., 2026) score supported criteria before and after each turn. We subtract the earlier score from the later one, so increases indicate progress and decreases regression. If either score is unavailable, we emit no local signal. Missing observations are never treated as zero completion. Checks run on recorded states without intermediate Judge calls.

Assigning turn credit. At least half of each positive increment stays with the turn that produced it. The rest is divided equally among earlier turns whose tool outputs or exact artifact versions were directly used to produce that progress, as confirmed by recorded dependencies. For example, a turn that reads a table may share credit when a later turn uses that returned table to compute a checked result. Merely opening a file does not qualify. If no earlier turn qualifies, all credit stays local. Negative increments remain entirely with the turn that produced them.

Policy update. We combine these allocations using normalized criterion weights $\bar{w}_{i,k}$ to obtain credit $\delta_{i,r,t}$ for turn t. We add this credit to the trajectory advantage and subtract a local trace cost $c_{i,r,t}$ for tool errors or unproductive repetition:

$$\widehat{A}_{i,r,t} = \text{sg}\left[A^{\text{traj}}_{i,r} + \alpha H_{i,r}\delta_{i,r,t} - \eta c_{i,r,t}\right], \qquad \alpha, \eta \geq 0, \tag{1}$$

Here α and η weight progress credit and local trace cost, and sg stops gradients. The gate $H_{i,r}$ is 1 only when integrity, protected-state, and safety checks pass. An unfinished rollout can retain progress credit, while violations disable it for every turn. Clipped GRPO (Shao et al., 2024) averages token contributions within each turn, then turns within each trajectory, so lengthy turns do not dominate the update. Setting $\alpha = \eta = 0$ recovers our episode-only baseline. The reliability of this shaping signal depends on checker coverage and dependency records (Appendix B.8).

4 Experiments

4.1 Experimental Setup

Model and training data. We post-train Qwen3.5-35B-A3B (Qwen Team, 2026a) to obtain WorkRover-35B-A3B. Supervised fine-tuning (SFT) uses 30,115 trajectories generated by

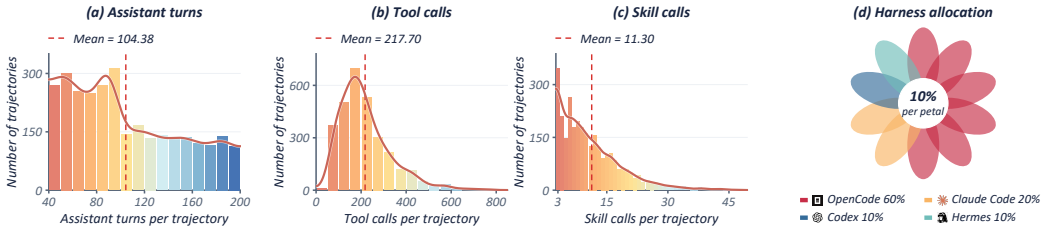

Figure 5: **Selected trajectory statistics.** (a–c) Assistant turns, tool calls, and skill calls in 3,200 selected successful traces. (d) Harness allocation.

Qwen3.5-397B-A17B in synthesized workspaces; all pass task verification and quality checks. For reinforcement learning (RL), we independently synthesize 20,000 task–Verifier pairs and screen each pair with four Qwen3.5-35B-A3B pilot rollouts. Consistency filtering retains 3,200 pairs for training. Starting from the SFT checkpoint, the model collects online rollouts on these tasks and learns with VTCA-GRPO (Sec. 3.3). Figure 5 summarizes 3,200 selected successful traces and the Harness allocation; subset-selection details are in Appendix C.1.

Evaluation. We evaluate transfer to five external benchmarks: JobBench (Li et al., 2026g) for occupational tasks, GDPval (Artificial Analysis, 2026c) for professional deliverables, AutomationBench (Shepard & Salimans, 2026) for cross-application workflows, WorkBuddy Bench (Office) (Tencent WorkBuddy Bench Team, 2026a) for office workflows, and OfficeQA Pro (Opsahl-Ong et al., 2026) for document-grounded question answering. We use each benchmark's official evaluator and aggregate metric, reporting GDPval in Elo and the other four scores as percentages. Table 1 compares WORKROVER with its backbone, other 27–35B open-weight models, open-weight models above 400B, and proprietary models. Published reference scores retain their original task subsets, evaluators, and harnesses. See details in Appendix C.1.

4.2 MAIN RESULTS

WORKROVER-35B-A3B improves on its Qwen3.5-35B-A3B backbone across all five external benchmarks (Table 1). Among the percentage-based scores, the largest gains are on OfficeQA Pro (+29.32 percentage points) and JobBench (+26.27 points), followed by AutomationBench (+22.67 points) and WorkBuddy Bench (Office) (+13.63 points). GDPval v2 increases from 804.75 to 1218.00 Elo, a gain of 413.25. These improvements span document reasoning, occupational tasks, professional deliverables, and application workflows, supporting transfer beyond the synthesized training workspaces.

Among the listed 27–35B open-weight models, WORKROVER has the highest reported score on every benchmark. Occamy-1.0 has the second-highest reported score in this group on each benchmark. The largest percentage-point gap between the two models is on WorkBuddy Bench (Office): 74.13% versus 66.65%, a margin of 7.48 points. The JobBench margin is 2.62 points (36.89% versus 34.27%), while the leads on AutomationBench and OfficeQA Pro are smaller, at 1.07 and 0.77 points, respectively.

Compared with larger and proprietary models, WORKROVER's relative performance varies across benchmarks. It exceeds GLM-5.2 and MiniMax-M3 on both AutomationBench and OfficeQA Pro, and Kimi K3 on WorkBuddy Bench (Office) (74.13% versus 72.18%). Gaps remain on occupational work and deliverable quality: its JobBench score of 36.89% is below GPT-5.5 (38.30%) and Kimi K3 (54.27%), and its GDPval score of 1218.00 Elo is below all listed open-weight models above 400B (1303.96–1572.42 Elo). WorkBuddy performance also remains below GPT-5.5 (86.05%) and Claude Opus 4.7 (92.51%). Comparisons with reference models use their reported scores under source-specific evaluation settings (Appendix C.1).

4.3 ABLATIONS AND ANALYSIS

4.3.1 VERIFIER-TO-TURN CREDIT ASSIGNMENT

With the SFT checkpoint, task and seed schedules, and rollout budget fixed, we compare VTCA with episode-only GRPO (E-GRPO; Figure 6). Both RL methods improve on SFT across all five benchmarks. VTCA further outperforms E-GRPO on every benchmark. The largest percentage-point gains are on AutomationBench (+5.7) and OfficeQA Pro (+5.4), while GDPval v2 improves by 43 Elo. These results support giving individual turns feedback beyond the final trajectory outcome. Protocol and component controls are in Appendices C.5 and B.8.

Table 1: Benchmark results. **Bold** and underlined mark the best and second-best scores per model group. Subscripts show changes from Qwen3.5-35B-A3B: ↑ increases and ↓ decreases (Elo or percentage points). Higher scores are better. Details in Appendix C.1.

Model	JobBench (%) ↑	GDPval (Elo) ↑	Automation Bench (%) ↑	WorkBuddy Bench (Office) (%) ↑	OfficeQA Pro (%) ↑
Proprietary Models					
GPT-5.5	38.30$_{\uparrow 27.68}$	**1395.85**$_{\uparrow 591.10}$	22.70$_{\uparrow 16.70}$	86.05$_{\uparrow 25.55}$	**57.10**$_{\uparrow 37.55}$
Claude Opus 4.7	**44.50**$_{\uparrow 33.88}$	1395.78$_{\uparrow 591.03}$	9.90$_{\uparrow 3.90}$	**92.51**$_{\uparrow 32.01}$	46.60$_{\uparrow 27.05}$
Gemini 3.1 Pro Preview	15.90$_{\uparrow 5.28}$	903.94$_{\uparrow 99.19}$	**37.54**$_{\uparrow 31.54}$	–	39.10$_{\uparrow 19.55}$
Open-Weight Models (>400B total parameters)					
Kimi K3	**54.27**$_{\uparrow 43.65}$	**1572.42**$_{\uparrow 767.67}$	**46.67**$_{\uparrow 40.67}$	72.18$_{\uparrow 11.68}$	63.30$_{\uparrow 43.75}$
DeepSeek-V4-Pro	54.10$_{\uparrow 43.48}$	1493.31$_{\uparrow 688.56}$	31.80$_{\uparrow 25.80}$	78.71$_{\uparrow 18.21}$	**65.40**$_{\uparrow 45.85}$
GLM-5.2	43.40$_{\uparrow 32.78}$	1417.62$_{\uparrow 612.87}$	26.17$_{\uparrow 20.17}$	**79.57**$_{\uparrow 19.07}$	41.40$_{\uparrow 21.85}$
MiniMax-M3	–	1303.96$_{\uparrow 499.21}$	15.90$_{\uparrow 9.90}$	76.30$_{\uparrow 15.80}$	45.10$_{\uparrow 25.55}$
Open-Weight Models (27–35B total parameters)					
Qwen3.6-27B	21.80$_{\uparrow 11.18}$	1068.87$_{\uparrow 264.12}$	9.67$_{\uparrow 3.67}$	65.65$_{\uparrow 5.15}$	44.36$_{\uparrow 24.81}$
Granite-4.2-30B	1.64$_{\downarrow 8.98}$	721.01$_{\downarrow 83.74}$	0.02$_{\downarrow 5.98}$	54.18$_{\downarrow 6.32}$	12.03$_{\downarrow 7.52}$
Muse-Glimmer-30B	11.82$_{\uparrow 1.20}$	892.72$_{\uparrow 87.97}$	6.83$_{\uparrow 0.83}$	57.45$_{\downarrow 3.05}$	24.06$_{\uparrow 4.51}$
Nex-N2-mini	3.82$_{\downarrow 6.80}$	1097.03$_{\uparrow 292.28}$	5.70$_{\downarrow 0.30}$	25.87$_{\downarrow 34.63}$	26.32$_{\uparrow 6.77}$
Occamy-1.0	34.27$_{\uparrow 23.65}$	1128.00$_{\uparrow 323.25}$	27.60$_{\uparrow 21.60}$	66.65$_{\uparrow 6.15}$	48.10$_{\uparrow 28.55}$
Qwen3.5-35B-A3B	10.62	804.75	6.00	60.50	19.55
WorkRover-35B-A3B (Ours)	**36.89**$_{\uparrow 26.27}$	**1218.00**$_{\uparrow 413.25}$	**28.67**$_{\uparrow 22.67}$	**74.13**$_{\uparrow 13.63}$	**48.87**$_{\uparrow 29.32}$

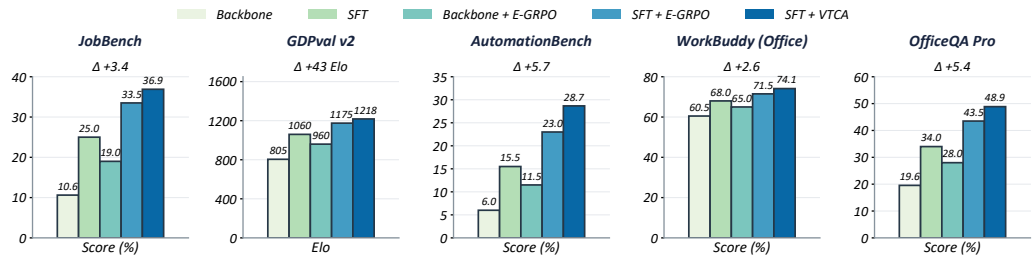

Figure 6: 35B-A3B comparison under matched rollout budgets. Backbone denotes Qwen3.5-35B-A3B before our SFT or RL. Gains compare SFT + VTCA with SFT + E-GRPO. GDPval v2 uses Elo; other scores are percentages.

4.3.2 Recursive Environment Evolution

We compare repair methods using Qwen3.5-35B-A3B under matched proposal and replay budget caps. Safe Resolution Rate (SRR) measures how often a defect is fixed without causing new failures in related tasks; repairs must also pass fresh-seed confirmation and a held-out audit. FCR raises SRR on naturally occurring defects from 60.7% with full regeneration to 70.3% (Figure 7(b)). It also reduces changes to task intent from 14.3% to 1.3% while using fewer model tokens (relative cost: 1.00 versus 1.34; Appendix Table 30).

Choosing compatible edits. With repair proposals fixed, Table 2 tests selection separately from proposal generation. Compared with applying all edits (Apply-all), FCR raises SRR by 9.0 percentage points on natural defects and 10.4 points on injected defects, and reduces cross-task regression from 7.0% to 0.7%. Testing edit combinations thus improves repair reliability, at a higher model-token cost (1.00 versus 0.59).

Joint repair and repeated evolution. With proposals fixed, 48–74% of safely resolved cases require joint edits, depending on the component pair: no smaller subset of the selected edits passes the same repair checks (Figure 7(a)). Across three generations, SRR stays at 70.3–72.4% and audit acceptance at 93.8–94.6% (panel c), showing little change in repair quality across these rounds. FCR's rollout success also increases over five repair iterations (panel d).

Repairing initial candidates. FCR also improves initial construction. With the candidate set and repair budget caps fixed, synthesis pass rate reaches 74.8%, compared with 49.8% before repair and 68.8% with full regeneration (Appendix Table 29). Repair therefore recovers more tasks that pass validation from the same initial pool.

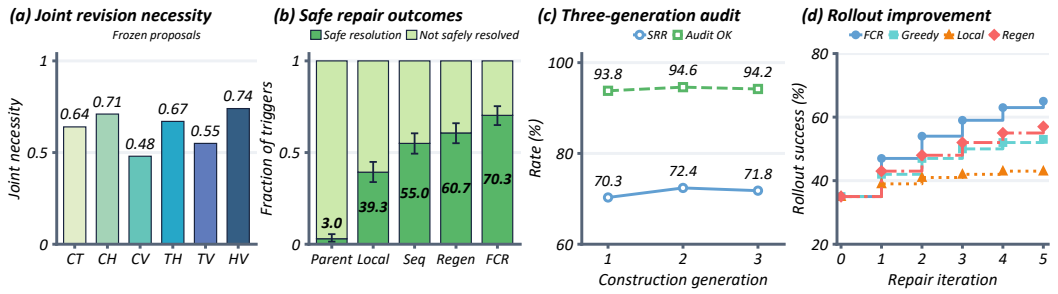

Figure 7: **FCR mechanism, stability, and evolution.** (a) Pairwise joint necessity (C: Context; T: Task; H: Harness; V: Verifier). (b) Natural-trigger outcomes. (c) SRR and sealed-audit acceptance across three generations. (d) Measured rollout success over five repair iterations.

4.3.3 Construction Pipeline

We remove one construction component at a time (Table 3). Task yield measures the fraction of initial candidates that pass validation. The largest drops come from removing relation-aware expansion (74.8% to 52.6%) or validation-guided rewriting (74.8% to 49.8%). These results highlight the value of following file dependencies and repairing construction errors.

Reference execution and agent success decline as well. Removing relation-aware expansion lowers them from 95.1% and 58.7% to 77.9% and 41.8%, respectively. Removing file–skill grounding also reduces both measures, supporting the need to match each file with operations that can use it. The semantic-only baseline performs worse on all three measures, suggesting that topical similarity alone is insufficient for assembling executable workspaces. Success rates use the same frozen agent but are measured on each variant's retained workspaces, so they can also reflect differences in which tasks are admitted.

Table 2: Maintenance with shared frozen proposals. Matched budget caps; rates (%); model-token cost relative to FCR. Intervals: Table 30.

Method	Natural SRR↑	Controlled SRR↑	Task reg.↓	Rel. cost↓
Local-only	42.0	45.4	3.2	0.43
Greedy-sequential	58.0	63.1	2.7	0.78
Apply-all	64.0	70.8	7.0	0.59
FCR	**73.0**	**81.2**	**0.7**	1.00

Table 3: Construction ablations (%). Mean ± workspace-clustered 95% interval half-width. Full variants: Table 31.

Construction	Task yield	Reference exec.	Success Rate
WorkRover (full)	$74.8_{\pm 2.20}$	$95.1_{\pm 1.10}$	$58.7_{\pm 2.60}$
w/o Job–Skill	$67.2_{\pm 2.38}$	$91.7_{\pm 1.48}$	$53.1_{\pm 2.78}$
w/o File–Skill	$60.9_{\pm 2.47}$	$84.2_{\pm 2.06}$	$47.4_{\pm 2.92}$
w/o Relation Expansion	$52.6_{\pm 2.53}$	$77.9_{\pm 2.52}$	$41.8_{\pm 3.11}$
w/o Validation Rewrite	$49.8_{\pm 2.53}$	$80.6_{\pm 2.47}$	$44.5_{\pm 3.22}$
Semantic-only	$48.7_{\pm 2.53}$	$73.6_{\pm 2.78}$	$38.6_{\pm 3.19}$

Rewriting on the same initial tasks. To control for differences in task selection, we compare filtering with and without rewriting on the same 10,000 initial task intents (Appendix Table 32). Both variants preserve the required outcomes and use the same frozen agent. Adding rewriting raises admission from 49.8% to 74.8% and end-to-end success from 22.2% to 43.9%. Reference execution also rises from 80.0% to 92.0%, and Verifier validity from 70.0% to 87.0%. Unconstructed, rejected, and budget-exhausted intents remain in the denominator and contribute no successful completion. The resulting gains therefore measure the full pipeline, from initial construction through agent execution, on the same starting task set.

5. Conclusion

WorkRover uses execution feedback to improve both long-horizon cowork agents and the environments used to train them. The Harness keeps work contexts, tasks, and verifiers aligned during construction and repair. FCR revises inconsistent components while preserving task requirements, and VTCA uses verified workspace progress to guide learning at individual turns. Experiments show that FCR increases the share of tasks that pass validation and repairs defects with fewer unintended changes to task intent and fewer regressions on related tasks. Under matched rollout budgets, VTCA outperforms episode-only GRPO on all five benchmarks. These findings support recursive environment evolution as a practical foundation for reliable long-horizon cowork agent training.

AI USE STATEMENT

We used AI tools to assist with writing and polishing the manuscript. Details of AI use in this work are described in the paper.

REFERENCES

Accio-Lab. Occamy-1.0 Model Card, 2026. URL https://huggingface.co/Accio-Lab/occamy-1.0. Occamy-1.0 and Nex-N2-mini benchmark results; accessed September 16, 2026.

Accio Team. Occamy-1.0: Open pareto-frontier 35b intelligence for co-work, 2026. URL https://arxiv.org/abs/2609.11977. Appendix B.3: benchmark-specific evaluation protocols.

Anthropic. Claude Cowork: Anthropic's agentic AI for knowledge work, 2026a. URL https://claude.com/product/cowork. Accessed: 2026-09-06.

Anthropic. Introducing Claude Opus 4.7, April 2026b. URL https://www.anthropic.com/news/claude-opus-4-7. Published April 16, 2026; accessed September 11, 2026.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/16105fb9cc614fc29e1bda00dab60d41-Abstract.html.

Artificial Analysis. AutomationBench-AA: Evaluation methodology, 2026a. URL https://artificialanalysis.ai/methodology/intelligence-benchmarking#automationbench-aa. Private 657-task dataset v1.0; objective-completion scoring; accessed September 6, 2026.

Artificial Analysis. AutomationBench-AA leaderboard, 2026b. URL https://artificialanalysis.ai/evaluations/automationbench-aa. Qwen3-8B, Qwen3-14B, Qwen3.6-27B, Qwen3.8-27B, and Muse-Glimmer-30B results; objective-completion scoring on 657 tasks; accessed September 8–9, 2026.

Artificial Analysis. GDPval v2 leaderboard, 2026c. URL https://artificialanalysis.ai/evaluations/gdpval-aa. GDPval v2 reference scores frozen to the September 12, 2026 snapshot (Asia/Shanghai); model configurations and source records documented in Appendix C.1.

Artificial Analysis. GLM-5.2 (max): Independent evaluation results, 2026d. URL https://artificialanalysis.ai/models/glm-5-2. GDPval v2 and AutomationBench-AA; accessed 2026-09-07.

Artificial Analysis. Kimi K3 (max): Independent evaluation results, 2026e. URL https://artificialanalysis.ai/models/kimi-k3. GDPval v2 and AutomationBench-AA; accessed 2026-09-07.

Artificial Analysis. MiniMax-M3: Independent evaluation results, 2026f. URL https://artificialanalysis.ai/models/minimax-m3. GDPval v2 and AutomationBench-AA; accessed 2026-09-07.

Yonas Atinafu and Robin Cohen. RewardHackingAgents: Benchmarking evaluation integrity for LLM ML-engineering agents, 2026. URL https://arxiv.org/abs/2603.11337.

Fei Bai, Huatong Song, Shuang Sun, Daixuan Cheng, Yike Yang, Chuan Hao, Renyuan Li, Feng Chang, Yuan Wei, Ran Tao, Bryan Dai, Jian Yang, Wayne Xin Zhao, and Ji-Rong Wen. ClawGym: A scalable framework for building effective Claw agents, 2026. URL https://arxiv.org/abs/2604.26904.

Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, Hang He, and Wei Lin. RLFactory: A plug-and-play reinforcement learning post-training framework for LLM multi-turn tool-use, 2025. URL https://arxiv.org/abs/2509.06980.

Hao Chen, Zhexin Hu, Jiajun Chai, Haocheng Yang, Hang He, Xiaohan Wang, Wei Lin, Luhang Wang, Guojun Yin, and Zhuofeng Zhao. ToolForge: A data synthesis pipeline for multi-hop search without real-world APIs, 2025. URL https://arxiv.org/abs/2512.16149.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. SkillCraft: Can LLM agents learn to use tools skillfully?, 2026. URL https://arxiv.org/abs/2603.00718.

Zelei Cheng, Amritansh Mishra, Sambit Sahu, and William Campbell. Efficient reinforcement learning for long-horizon tool-use agentic tasks, 2026a. URL https://arxiv.org/abs/2608.10357.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-World: Scaling terminal-agent environments via agent skills, 2026b. URL https://arxiv.org/abs/2605.20876.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025. URL https://arxiv.org/abs/2505.22617.

Databricks AI Research. OfficeQA Pro: Official agent harness performance results. Official benchmark repository, 2026. URL https://github.com/databricks/officeqa#officeqa-pro--agent-harness-performance. Accessed September 8, 2026.

DeepSeek-AI. DeepSeek-V4-Pro-0813 Model Card: Public AutomationBench Comparison, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro-0813. August 13 official release; AutomationBench Public score 31.8; accessed September 12, 2026.

Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Yang JingYi, Penghui Yang, Zhixiong Zhang, Xilin Wei, Xinyu Fang, Yubo Ma, Haodong Duan, Jing Shao, Jiaqi Wang, Dahua Lin, Kai Chen, and Yuhang Zang. WildClawBench: A benchmark for real-world, long-horizon agent evaluation, 2026. URL https://arxiv.org/abs/2605.10912.

Chengqi Dong, Chuhuai Yue, Hang He, Yandong Liu, Fenghe Tang, S Kevin Zhou, Xiaohan Wang, Jiajun Chai, and Guojun Yin. TAPO: Tool-aware policy optimization via credit transfer for multimodal search agents, 2026a. URL https://arxiv.org/abs/2606.05784.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. AgentWorld: Scaling real-world environment synthesis for evolving general agent intelligence, 2026b. URL https://arxiv.org/abs/2604.18292.

Haoyu Dong, Pengkun Zhang, Yan Gao, Xuanyu Dong, Yilin Cheng, Mingzhe Lu, Zikun Zhu, Adina Yakefu, and Shuxin Zheng. Finch: Benchmarking finance & accounting across spreadsheet-centric enterprise workflows, 2025. URL https://arxiv.org/abs/2512.13168.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks?, 2024. URL https://arxiv.org/abs/2403.07718.

Esakkivel Esakkiraja, Denis Akhiyarov, Vikas Yadav, Sai Rajeswar, Patrice Bechard, Sridhar Nemala, and Sagar Davasam. StarHarness: Evolving harnesses with stratified search for enterprise environments, 2026. URL https://arxiv.org/abs/2608.24804v1.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, and Lilin Wang. Toward scalable terminal task synthesis via skill graphs, 2026. URL https://arxiv.org/abs/2604.25727.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Jonathan Gabor, Jayson Lynch, and Jonathan Rosenfeld. EvilGenie: A reward hacking benchmark, 2025. URL https://arxiv.org/abs/2511.21654.

Shubham Gandhi, Saurabh Goyal, Kiran Kate, and Yara Rizk. DRACO: Fine-grained credit assignment with dynamic rubrics for long-horizon agent training, 2026. URL https://arxiv.org/abs/2609.04094.

Google DeepMind. Gemini 3.1 Pro model card, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro. Published February 19, 2026; accessed September 11, 2026.

Yupu Hao, Zhuoran Jin, Huanxuan Liao, Kang Liu, and Jun Zhao. Why multi-step tool-use reinforcement learning collapses and how supervisory signals fix it, 2026. URL https://arxiv.org/abs/2606.26027.

Hang He, Chuhuai Yue, Chengqi Dong, Mingxue Tian, Hao Chen, Zhenfeng Liu, Jiajun Chai, Xiaohan Wang, Yufei Zhang, Qun Liao, Guojun Yin, Wei Lin, Chengcheng Wan, Haiying Sun, and Ting Su. LocalSearchBench: Benchmarking agentic search in real-world local life services. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026a. doi: 10.1145/3770855.3817466. URL https://doi.org/10.1145/3770855.3817466.

Hang He, Chuhuai Yue, Chengqi Dong, Chengcheng Wan, Ting Su, Haiying Sun, Jiajun Chai, Xiaohan Wang, and Guojun Yin. VistaHop: Benchmarking Long-Horizon Visual DeepSearch, 2026b. URL https://arxiv.org/abs/2606.03273.

Hert4. AutomationBench 600: Open-Weight Reproduction, 2026. URL https://github.com/Hert4/automationbench-opera-gemma4. Public v1.0.5; Qwen3.6-27B 58/600; accessed September 12, 2026.

Yining Hua, Hongbin Na, Yifan Zhou, Akshay Kalose, Cyrus Ayubcha, and Levi Lian. StagedWorkspace: A versioned workspace for knowledge-work agents, 2026a. URL https://arxiv.org/abs/2608.18050.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, Letian Zhu, Xinping Lei, Han Li, Zhiyuan Ma, Zili Wang, Zhaoxiang Zhang, and Jiaheng Liu. CLI-Universe: Towards verifiable task synthesis engine for terminal agents, 2026b. URL https://arxiv.org/abs/2606.22883.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, Yifei Ming, Jiaqi Pan, Bhavana Dalvi Mishra, Jiaxin Huang, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. EnvHarness: Awakening static worlds for agent learning, 2026. URL https://arxiv.org/abs/2608.19880.

IBM Granite Team. Granite-4.2-30B model card, August 2026. URL https://huggingface.co/ibm-granite/granite-4.2-30b. Official 30B dense open-weight release under Apache 2.0; accessed September 9, 2026.

Maksim Ivanov and Abhijay Rana. Anchor: Mitigating artifact drift in agent benchmark generation, 2026. URL https://arxiv.org/abs/2605.26321.

Minbyul Jeong and Chanwoong Yoon. AgentMercury: Your agent can synthesize verifiable environments for business scenarios at scale, 2026. URL https://arxiv.org/abs/2608.20634.

Haonian Ji, Kaiwen Xiong, Siwei Han, Peng Xia, Shi Qiu, Yiyang Zhou, Jiaqi Liu, Jinlong Li, Bingzhou Li, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. ClawArena: Benchmarking AI agents in evolving information environments, 2026. URL https://arxiv.org/abs/2604.04202.

Rui Jin, Jikai Chen, Yihan Chen, Hao Zhou, Demin Zhu, Kaichen Yang, Dong Wang, Linjian Mo, and Chenyi Zhuang. Rubric-to-code credit assignment for reinforcement learning, 2026. URL https://arxiv.org/abs/2608.27906.

JobBench Team. JobBench: Official leaderboard, 2026. URL https://job-bench.github.io/. JobBench-Main; OpenCode v1.14.18 and Grok 4.3; accessed September 6–7, 2026.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Will LeVine, Brendan Evers, Sam Saltwick, and Abhay Venkatesh. RubricRefine: Improving tool-use agent reliability with training-free pre-execution refinement, 2026. URL https://arxiv.org/abs/2605.09730.

Boyan Li, Zhuowen Liang, Yupeng Xie, Xiaotian Lin, Tianqi Luo, Xinyu Liu, Yizhang Zhu, Zhangyang Peng, Yuan Li, Zhengxuan Zhang, Jiayi Zhang, Nan Tang, Guoliang Li, and Yuyu Luo. DataSpace: Benchmarking data agents for verifiable analytics over heterogeneous workspaces, 2026a. URL https://arxiv.org/abs/2608.03451.

Chenxin Li, Zhengyang Tang, Mingxin Huang, Yunlong Lin, Shijue Huang, Shengyuan Liu, Bowen Ye, Rang Li, Lei Li, Benyou Wang, and Yixuan Yuan. Claw-Eval-Live: A live agent benchmark for evolving real-world workflows, 2026b. URL https://arxiv.org/abs/2604.28139.

Gang Li, Ming C. Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. DisCO: Reinforcing large reasoning models with discriminative constrained optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1921. URL https://papers.neurips.cc/paper_files/paper/2025/hash/52d6c85793b3dd47636eb430710da915-Abstract-Conference.html.

Jinchao Li, Yunxin Li, Chenrui Zhao, Zhenran Xu, Baotian Hu, and Min Zhang. WindowsWorld: A process-centric benchmark of autonomous GUI agents in professional cross-application environments, 2026c. URL https://arxiv.org/abs/2604.27776.

Xiangyi Li, Kyoung Whan Choe, Yimin Liu, Xiaokun Chen, Chujun Tao, Bingran You, Wenbo Chen, Zonglin Di, Jiankai Sun, Shenghan Zheng, Jiajun Bao, Yuanli Wang, Weixiang Yan, Yiyuan Li, and Han-chung Lee. ClawsBench: Evaluating capability and safety of LLM productivity agents in simulated workspaces, 2026d. URL https://arxiv.org/abs/2604.05172.

Xirui Li, Ming Li, Ion Stoica, Cho-Jui Hsieh, and Tianyi Zhou. ClawEnvKit: Automatic environment generation for claw-like agents, 2026e. URL https://arxiv.org/abs/2604.18543.

Yuanyang Li, Xue Yang, Longyue Wang, Weihua Luo, and Hongyang Chen. ComplexMCP: Evaluation of LLM agents in dynamic, interdependent, and large-scale tool sandbox, 2026f. URL https://arxiv.org/abs/2605.10787.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, Xinyang Han, Brian Lee, Kayla Xu, Shenglai Zeng, Hang Hua, Xiangliang Zhang, Basel Alomair, Ranjay Krishna, Luke Zettlemoyer, Pang Wei Koh, Bhaskar Ramasubramanian, Luyao Niu, Xiang Yue, and Radha Poovendran. JobBench: Aligning agent work with human will, 2026g. URL https://arxiv.org/abs/2605.26329.

Zhiwei Li and Yong Hu. SkillHone: A harness for continual agent skill evolution through persistent decision history, 2026. URL https://arxiv.org/abs/2606.08671.

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

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-driven automatic evolution of coding-agent harnesses, 2026a. URL https://arxiv.org/abs/2604.25850.

Yin Lin, Elaine Ang, Erkang Zhu, Bolin Ding, and Jingren Zhou. Context as an environment: Programmatic context management for long-horizon agents, 2026b. URL https://arxiv.org/abs/2608.21690.

Shuai Liu, Shulin Tian, Kairui Hu, Yuhao Dong, Zhe Yang, Bo Li, Jingkang Yang, Chen Change Loy, and Ziwei Liu. FileGram: Grounding agent personalization in file-system behavioral traces, 2026a. URL https://arxiv.org/abs/2604.04901.

Yimeng Liu, Misha Sra, Jeevana Priya Inala, and Chenglong Wang. ReUseIt: Synthesizing reusable AI agent workflows for web automation. In *Proceedings of the 31st International Conference on Intelligent User Interfaces*. Association for Computing Machinery, 2026b. doi: 10.1145/3742413.3789083. URL https://doi.org/10.1145/3742413.3789083.

Yue Liu, Zhiyuan Hu, Flood Sung, Jiaheng Zhang, and Bryan Hooi. KLong: Training LLM agent for extremely long-horizon tasks, 2026c. URL https://arxiv.org/abs/2602.17547.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03680.

Ziyu Ma, Hailang Huang, Shun Zou, Yong Wang, Shidong Yang, Yiming Hu, Fei Wei, and Xiangxiang Chu. LongHorizon-Harness: Advancing long-horizon agents for real-world tasks, 2026. URL https://arxiv.org/abs/2608.01964.

Shiva Krishna Reddy Malay, Shravan Nayak, Jishnu Sethumadhavan Nair, Sagar Davasam, Aman Tiwari, Sathwik Tejaswi Madhusudhan, Sridhar Krishna Nemala, Srinivas Sunkara, and Sai Rajeswar. EnterpriseOps-Gym: Environments and evaluations for stateful agentic planning and tool use in enterprise settings, 2026. URL https://arxiv.org/abs/2603.13594.

Fanqing Meng, Lingxiao Du, Zijian Wu, Guanzheng Chen, Xiangyan Liu, Jiaqi Liao, Chonghe Jiang, Zhenglin Wan, Jiawei Gu, Pengfei Zhou, Rui Huang, Ziqi Zhao, Shengyuan Ding, Ailing Yu, Bo Peng, Bowei Xia, Hao Sun, Haotian Liang, Ji Xie, Jiajun Chen, Jiajun Song, Liu Yang, Ming Xu, Qionglin Qiu, Runhao Fu, Shengfang Zhai, Shijian Wang, Tengfei Ma, Tianyi Wu, Weiyang Jin, Yan Wang, Yang Dai, Yao Lai, Youwei Shu, Yue Liu, Yunzhuo Hao, Yuwei Niu, Jinkai Huang, Jiayuan Zhuo, Zhennan Shen, Linyu Wu, Hannah Yao, Charles Chen, Cihang Xie, Yuyin Zhou, Jiaheng Zhang, Zeyu Zheng, Mengkang Hu, and Michael Qizhe Shieh. ClawMark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents, 2026. URL https://arxiv.org/abs/2604.23781.

Meta Models. Muse-Glimmer-30B model card, August 2026. URL https://huggingface.co/meta-models/Muse-Glimmer-30B. Official approximately 30B dense open-weight release under Apache 2.0; accessed September 9, 2026.

Microsoft. Microsoft 365 Copilot Cowork: Overview, 2026. URL https://learn.microsoft.com/en-us/microsoft-365/copilot/cowork/. Accessed: 2026-09-06.

MiniMax. MiniMax M3: Frontier coding, 1M context, native multimodality—all in one model, June 2026. URL https://www.minimax.io/blog/minimax-m3. Published June 1, 2026; accessed September 8, 2026.

Moonshot AI. Kimi K3 model card, 2026. URL https://huggingface.co/moonshotai/Kimi-K3. Accessed September 8, 2026.

Nex AGI. Nex-N2-mini model card, 2026. URL https://huggingface.co/nex-agi/Nex-N2-mini. Accessed September 11, 2026.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287. Morgan Kaufmann, 1999. URL https://ai.stanford.edu/~ang/papers/shaping-icml99.pdf.

Zechun Niu, Yukun Zhao, Jiaxin Zhang, Xu Shen, Jinhua Si, Han Tian, Can Xu, Yunfan Song, Jiaxin Mao, Yansong Gao, Yuchen Li, Jianmin Wu, Lingyong Yan, Shuaiqiang Wang, and Dawei Yin. DuMateBench: Evaluating autonomous agents in complex real-world workflows, 2026. URL https://arxiv.org/abs/2608.26546.

OpenAI. GPT-5.5 system card, April 2026. URL https://openai.com/index/gpt-5-5-system-card/. Published April 23, 2026; accessed September 11, 2026.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL https://arxiv.org/abs/2603.08655.

Tianjun Pan, Xuan Lin, Wenyan Yang, Qianyu He, Shisong Chen, Licai Qi, Wanqing Xu, Hongwei Feng, Bo Xu, and Yanghua Xiao. RubricEval: A rubric-level meta-evaluation benchmark for LLM judges in instruction following, 2026a. URL https://arxiv.org/abs/2603.25133.

Zhihong Pan, Jiyuan He, Kai Zhang, Yupeng Han, Ze Liu, Yuze Zhao, Yongcong Ye, and Zhaohua Yang. Meta-Task: Turning terminal task synthesis into a terminal task for scalable agent training, 2026b. URL https://arxiv.org/abs/2607.27929.

Yash Pandya, Sahil Gupta, Sarthak Harne, Archana Yadav, Kavyansh Chourasia, Hussein Mozannar, Vibhav Vineet, Sara Abdali, Corby Rosset, Yash Lara, Ahmed Awadallah, Ece Kamar, and Akshay Nambi. Echoverse: Deep, evolving environments for training computer-use agents at scale, 2026. URL https://arxiv.org/abs/2607.28074.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks, 2025. URL https://arxiv.org/abs/2510.04374.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents, 2026a. URL https://arxiv.org/abs/2602.16165.

Yangda Peng, Yunjia Qi, Haotian Xia, Guanzhong He, Xintong Shi, Richeng Xuan, Songyuanyi Lu, Yixian Liu, Zhichao Hu, Yuhong Liu, and Hao Peng. Can LLM-as-a-judge reliably verify rubrics in agentic scenarios?, 2026b. URL https://arxiv.org/abs/2606.29920.

Wasu Top Piriyakulkij, Rachel Lawrence, Alicia Curth, Sushrut Karmalkar, and Niranjani Prasad. Subagents vs agent skills: Executing reusable knowledge for long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2609.09233.

Qoder. QoderWork: A desktop agentic assistant for everyday workflows, 2026. URL https://docs.qoder.com/qoderwork/introduction. Accessed: 2026-09-06.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5. Official citation recommended by the Qwen3.5-35B-A3B model card; accessed September 11, 2026.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27b dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b. Official model release; weights at https://huggingface.co/Qwen/Qwen3.6-27B; accessed September 9, 2026.

Qwen Team. Qwen3.8-27B model card, 2026c. URL https://huggingface.co/Qwen/Qwen3.8-27B. Self-reported JobBench scores: Qwen3.6-27B 21.8 and Qwen3.8-27B 33.4; accessed September 10, 2026.

Corby Rosset, Pratyusha Sharma, Andrew Zhao, Miguel Gonzalez-Fernandez, and Ahmed Awadallah. The art of building verifiers for computer use agents, 2026. URL https://arxiv.org/abs/2604.06240.

Amit Roth, Ankur Samanta, Matan Halevy, Yoav Levine, and Yonathan Efroni. Hack-verifiable environments: Towards evaluating reward hacking at scale, 2026. URL https://arxiv.org/abs/2605.20744.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Leyang Shen, Yang Zhang, Chun Kai Ling, Xiaoyan Zhao, and Tat-Seng Chua. CARL: Criticality-aware agentic reinforcement learning, 2025. URL https://arxiv.org/abs/2512.04949.

Daniel Shepard and Robin Salimans. AutomationBench, 2026. URL https://arxiv.org/abs/2604.18934.

Kou Shi, Zun Wang, Qisheng Su, Shiting Huang, Ziao Zhang, Zhen Fang, Qingnan Ren, Jin Liu, Yu Zeng, Yiming Zhao, Lin Chen, Zehui Chen, and Feng Zhao. FACET: Preserving source intent and executable state in terminal task synthesis, 2026. URL https://arxiv.org/abs/2608.18580.

Huatong Song, Fei Bai, Ming Yang, Renyuan Li, Jia Deng, Jujie He, Zhange Zhang, Daixuan Cheng, Yan Xing, Qi Yun, Xuxing Chen, Danyang Li, Feng Chang, Chuan Hao, Ran Tao, Jian Yang, Bryan Dai, Wayne Xin Zhao, Mingjie Tang, and Ji-Rong Wen. ClawGym II: Exploring black-box RL on agent harness, 2026a. URL https://arxiv.org/abs/2608.16798.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis, 2026b. URL https://arxiv.org/abs/2601.05808.

Haoyang Su and Ying Wen. Learning CLI agents with structured action credit under selective observation, 2026. URL https://arxiv.org/abs/2605.08013.

Jingyu Sun, Yuyang Xue, Mingyang Li, Zhengtao Yao, Jiachen Li, Yang Cui, Wenhao Cai, Haozhe Liu, Fangying Wang, Magdalene Katharina Montgomery, Syed Murtuza Baker, and Hongpeng Zhou. TrajWiki: Source-grounded memory trajectories for long-horizon dialogue agents, 2026. URL https://arxiv.org/abs/2608.00967.

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

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents, 2026a. URL https://arxiv.org/abs/2603.08754.

Zelin Tan, Yiqun Zhang, Hao Li, Zhiyao Cui, Hejia Geng, Shao Zhang, Hangfan Zhang, Yang Chen, Xiaosong Wang, Lilong Wang, Zhenfei Yin, Shuyue Hu, Chen Zhang, and Lei Bai. SKT: Skill-use training at scale via verified synthetic data generation, 2026b. URL https://arxiv.org/abs/2608.02287.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, Shaoli Yu, Jinqi Wang, Zihang Zhou, Hongyi Zhou, Yuting Lv, Jinyang Li, Jiashuo Liu, Ruoyu Chen, Chunwei Liu, GuoLiang Li, Jihua Kang, and Fan Wu. Workspace-Bench 1.0: Benchmarking AI agents on workspace tasks with large-scale file dependencies, 2026. URL https://arxiv.org/abs/2605.03596.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.13988.

Tencent. WorkBuddy: A full-scenario AI agent workstation, 2026. URL https://cloud.tencent.com/product/workbuddy. Accessed: 2026-09-06.

Tencent WorkBuddy Bench Team. Tencent WorkBuddy Bench: A multi-domain coding-agent benchmark with contamination-resistant task construction, 2026a. URL https://arxiv.org/abs/2607.20911.

Tencent WorkBuddy Bench Team. WorkBuddy Bench: Official leaderboard data, 2026b. URL https://workbuddybench.com/data.js?v=22. Historical Office v1.2 and the July 15, 2026 Office snapshot; CodeBuddy Code, think mode; accessed September 6–7, 2026.

Zihang Tian, Jingsen Zhang, Rui Li, Xiaohe Bo, Yuanzi Li, and Xu Chen. ARCO: Adaptive rubrics with co-evolution for multi-step LLM-based agents, 2026. URL https://arxiv.org/abs/2606.21262.

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.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents, 2026. URL https://arxiv.org/abs/2601.14242.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM agents via stepwise progress attribution, 2025. URL https://arxiv.org/abs/2505.20732.

Wenkai Wang, Tao Xiong, Jingchen Ni, Yunpeng Bao, Xiyun Li, Tianqi Liu, Hongcan Guo, Zilong Huang, and Shengyu Zhang. DeskCraft: Benchmarking desktop agents on professional workflows and human-in-the-loop collaboration, 2026a. URL https://arxiv.org/abs/2606.03103.

Xuehai Wang, Haowei Qin, Tongxin Liu, Junkai Li, Buqiang Xu, Jintian Zhang, Yijun Chen, Zirui Xue, and Shumin Deng. Learning to evaluate before improving: Automatic rubric induction for automatic research agents, 2026b. URL https://arxiv.org/abs/2608.31076.

Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In *Proceedings of the 31st International Conference on Software Engineering*, pp. 364–374. IEEE, 2009. doi: 10.1109/ICSE.2009.5070536. URL https://profsforrest.github.io/homepage/data/publications/2009-icse-genprog.pdf.

Jie Wu, Zhenru Zhang, Beichen Zhang, Xuwu Wang, Yuhui Su, Mouxiang Chen, Peng Wang, Zhihai Wang, Que Shen, Hao Zhou, An Yang, Fei Huang, Yujiu Yang, and Dayiheng Liu. TerminalUniverse: Turning agent trajectories into scalable terminal environments, 2026a. URL https://arxiv.org/abs/2609.04148.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Zhichao Shi, Hao Zhou, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Envs-FORGE: Frontier-optimized reward-grounded environment synthesis for agent RL, 2026b. URL https://arxiv.org/abs/2608.14312.

Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, Zirui Wang, Jinjie Ni, Yufan Yang, Arvin Xu, and Michael Qizhe Shieh. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use, 2025. URL https://arxiv.org/abs/2509.24002.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning, 2025b. URL https://arxiv.org/abs/2509.08755.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress, 2025c. URL https://arxiv.org/abs/2511.08325.

Zhiheng Xi, Dingwen Yang, Jiaqi Liu, Jixuan Huang, Honglin Guo, Baodai Huang, Tinggang Chen, Qi Zhang, Zhonghang Lu, Chenyu Liu, Jiajun Sun, Jiazheng Zhang, Dingwei Zhu, Xin Guo, Junzhe Wang, Zhihao Zhang, Yuming Yang, Junjie Ye, Minghe Gao, Dongrui Liu, Jiaming Ji, Guohao Li, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentGym2: Benchmarking large language model agents in de-idealized real-world environments, 2026. URL https://arxiv.org/abs/2607.05174.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Ke Xu, Han Xu, Xinran Chen, Yuqian Wang, Zhixuan Li, Xiaojian Liu, Changwo Wu, Jianqiang Xia, and Yuchen Li. STAMP: Provenance-guided credit assignment for deep search agents, 2026a. URL https://arxiv.org/abs/2607.11172.

Yuanda Xu, Zhengze Zhou, Hejian Sang, Xiaomin Li, Jiaxin Zhang, Xinchen Du, Sen Na, Zhipeng Wang, and Alborz Geramifard. TRIAGE: Role-typed credit assignment for agentic reinforcement learning, 2026b. URL https://arxiv.org/abs/2606.32017.

Shidong Yang, Ziyu Ma, Tongwen Huang, Yiming Hu, Yong Wang, and Xiangxiang Chu. CoEvolve: Training LLM agents via agent-data mutual evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23015–23036. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.acl-long.1055. URL https://aclanthology.org/2026.acl-long.1055/.

Shidong Yang, Ziyu Ma, Tongwen Huang, Xucong Wang, Renda Li, Yiming Hu, Yong Wang, and Xiangxiang Chu. SkillForge: Evolving verifiable skills for reinforcement learning agents, 2026b. URL https://arxiv.org/abs/2608.24747.

Xiao Yu, Baolin Peng, Ruize Xu, Hao Zou, Qianhui Wu, Hao Cheng, Wenlin Yao, Nikhil Singh, Zhou Yu, and Jianfeng Gao. OpenForgeRL: Train harness-native agents in any environment, 2026a. URL https://arxiv.org/abs/2607.21557.

Zhuoyun Yu, Xin Xie, Wuguannan Yao, Chenxi Wang, Lei Liang, Xiang Qi, and Shumin Deng. SkillAdaptor: Self-adapting skills for LLM agents from trajectories, 2026b. URL https://arxiv.org/abs/2606.01311.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lu, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks, 2026. URL https://arxiv.org/abs/2606.29537.

Z.ai. GLM-5.2 model card, 2026. URL https://huggingface.co/zai-org/GLM-5.2. Accessed September 11, 2026.

Zapier. AutomationBench Public Results, 2026. URL https://github.com/zapier/AutomationBench. 600 public tasks; strict pass rate; accessed September 12, 2026.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue (Steve) Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification, 2026a. URL https://arxiv.org/abs/2604.01687.

Huan Zhang, Mingju Chen, Dongxu Zhou, Can Lv, Heng Chang, Sen Cui, Faguo Wu, and Shiji Zhou. TRCA: Transition-wise rubric credit assignment for long-horizon LLM agents, 2026b. URL https://arxiv.org/abs/2608.16156.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners, 2023. URL https://arxiv.org/abs/2308.10144.

Jiarong Zhao, Zhikai Lei, Zhiheng Xi, Rui Zheng, Hang Yan, Jie Zhou, Qin Chen, and Liang He. NexForge: Scaling agent capabilities through requirement-driven task synthesis for LLMs, 2026. URL https://arxiv.org/abs/2607.14186v6. Version 6, August 1, 2026.

Qi Zhao, Haotian Fu, Chen Sun, and George Konidaris. EPO: Hierarchical LLM agents with environment preference optimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6401–6415. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.367. URL https://aclanthology.org/2024.emnlp-main.367/.

Jincheng Zhong, Weizhi Wang, Che Jiang, Kai Tian, Zhenzhao Yuan, Junlin Yang, Dianqiao Lei, and Kaiyan Zhang. EnterpriseClawBench: Benchmarking agents from real workplace sessions, 2026a. URL https://arxiv.org/abs/2606.23654.

Ziqian Zhong, Ivgeni Segal, Ivan Bercovich, Shashwat Saxena, Kexun Zhang, and Aditi Raghunathan. Hardening agent benchmarks with adversarial hacker-fixer loops, 2026b. URL https://arxiv.org/abs/2606.08960.

Chenyu Zhou, Qiliang Jiang, Shuning Wu, and Xu Zhou. Coverage, not targeting: A structural regime in multi-turn agent credit assignment, 2026a. URL https://arxiv.org/abs/2609.02417.

Jingbo Zhou, Yusai Zhao, Qi Bao, Jingjia Cao, Zhenghai Chen, Chang Gao, Kaiqi Guo, Muxin Guo, Mingxuan Li, Xinjiang Lu, Yanru Ma, Yixiong Xiao, Zenghui Zhang, Le Zhang, and Hua Wu. OmegaUse-OfficeVal: Benchmarking LLM agents on long-horizon office-suite tasks with economic grounding, 2026b. URL https://arxiv.org/abs/2607.27155.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.

Minghang Zhu, Chuyang Wei, Junhao Xu, Yilin Cheng, Zhumin Chen, and Jiyan He. DEEPRUBRIC: Evidence-tree rubric supervision for efficient reinforcement learning of deep research agents, 2026. URL https://arxiv.org/abs/2606.17029.

APPENDIX CONTENTS

A Extended Related Work 21

A.1 Professional Work in Persistent Environments 21

A.2 Executable Synthesis and Cross-Artifact Compatibility 21

A.3 Verifier Reliability Under Changing Workspace State 22

A.4 Environment Evolution and Repair Selection 22

A.5 Long-Horizon Learning and Progress Attribution 22

A.6 Criteria for the Method Comparison 23

B Additional Method Details 24

B.1 Work Context Construction Details 24

B.1.1 Job Role Inventory 24

B.1.2 Workspace Construction from Deep Research Sessions 26

B.1.3 Recorded Deep Research Examples 28

B.2 Task Synthesis and Quality Control 28

B.2.1 Task Design and Review 30

B.2.2 Recorded Task Example: Quarterly Executive Handoff 30

B.3 Full Verifier Specification and Calibration Protocol 31

B.4 Full FCR Mechanics and Promotion Rules 36

B.5 Evaluation Protocol for Recursive Environment Evolution 38

B.6 Protocol Summary Tables 43

B.7 Canonical Trajectory Normalization and Diagnostics 43

B.8 Verifier-to-Turn Credit Assignment Details 45

C Additional Experiments and Evaluation Details 51

C.1 Experimental Setup 51

C.2 Scale and Quality of Environment Synthesis 54

C.3 Task Executability and Verifier Reliability 55

C.4 Long-Horizon Cowork Agent Performance 56

C.5 Ablation Study of Verifier-to-Turn Credit Assignment 57

C.6 Analysis of Recursive Environment Evolution 59

C.7 Construction Ablations and Error Analysis 62

A Extended Related Work

Research on professional agents spans persistent context, executable task generation, environment maintenance, and learning from interaction evidence (Lin et al., 2026b; Sun et al., 2026; Shi et al., 2026; Pandya et al., 2026; Zhang et al., 2026b). We review these areas with attention to the units of synthesis and repair, their validation procedures, and the evidence used for local learning signals. Table 4 summarizes the mechanisms reported by selected methods and locates WorkRover's FCR and VTCA within this literature.

A.1 Professional Work in Persistent Environments

Evaluation requirements. Professional-work benchmarks define the demands on training environments. WebArena, WorkArena, OSWorld, and AgentGym 2.0 establish executable settings (Zhou et al., 2023; Drouin et al., 2024; Xie et al., 2024; Xi et al., 2026); OSWorld 2.0, WindowsWorld, and MCPMark extend interaction horizons and application/tool coverage (Yuan et al., 2026; Li et al., 2026c; Wu et al., 2025). Stateful enterprise services and reproducible tool execution are represented by EnterpriseOps-Gym, ClawsBench, ComplexMCP, WildClawBench, and Claw-Eval-Live (Malay et al., 2026; Li et al., 2026d;f; Ding et al., 2026; Li et al., 2026b).

ClawMark, Workspace-Bench, and ClawArena extend evaluation to changing conditions, cross-file dependencies, and belief revision across sessions (Meng et al., 2026; Tang et al., 2026; Ji et al., 2026). Workplace sessions, collaboration, and file-derived memory motivate EnterpriseClawBench, DuMateBench, DeskCraft, and FileGram (Zhong et al., 2026a; Niu et al., 2026; Wang et al., 2026a; Liu et al., 2026a).

Artifact-centered requirements span JobBench, Finch, WorkBuddy Bench, AutomationBench, GDPval, APEX-Agents, and OmegaUse-OfficeVal (Li et al., 2026g; Dong et al., 2025; Tencent WorkBuddy Bench Team, 2026a; Shepard & Salimans, 2026; Patwardhan et al., 2025; Vidgen et al., 2026; Zhou et al., 2026b). OfficeQA Pro and DataSpace emphasize document and analytics grounding (Opsahl-Ong et al., 2026; Li et al., 2026a). StagedWorkspace binds views and review diffs to file content hashes (Hua et al., 2026a). Claude Cowork, QoderWork, WorkBuddy, and Microsoft 365 Copilot Cowork illustrate deployment settings with these requirements (Anthropic, 2026a; Qoder, 2026; Tencent, 2026; Microsoft, 2026).

Workspace reuse. These benchmarks characterize complementary aspects of persistent work: recoverable execution, cross-file consistency, and outcome-based evaluation. WorkRover uses related requirements when constructing versioned task packages that share recoverable Work Contexts. Its maintenance procedure revalidates the packages affected by shared component changes.

A.2 Executable Synthesis and Cross-Artifact Compatibility

Task and environment generation. AppWorld, ToolForge, and EnvScaler provide complementary application, virtual-tool, and programmatic environment construction methods (Trivedi et al., 2024; Chen et al., 2025; Song et al., 2026b). ClawGym, ClawEnvKit, and NexForge materialize workspaces or services around task and capability requirements (Bai et al., 2026; Li et al., 2026e; Zhao et al., 2026).

Terminal-World, SKT, and SkillSynth use skills to build tasks and workflows (Cheng et al., 2026b; Tan et al., 2026b; Fan et al., 2026). Meta-Task and CLI-Universe emphasize executable validation (Pan et al., 2026b; Hua et al., 2026b). Recursive Synthesis extends verified seeds into longer tasks (Li et al., 2026h). SKT repairs failed packages; FACET grounds artifacts in shared state and repairs inconsistencies while preserving source intent; Envs-FORGE rewrites instructions, fixtures, solutions, tests, and environments together (Tan et al., 2026b; Shi et al., 2026; Wu et al., 2026b).

Shared environments also support reuse (Xi et al., 2025a). AgentMercury samples tasks from persistent business worlds, while Agent-World uses capability diagnostics to expand tool/database environments (Jeong & Yoon, 2026; Dong et al., 2026b). Terminal-Universe reconstructs workspaces from file operations and expands them into tasks and sessions with feedback and regression checks (Wu et al., 2026a).

Construction units. These approaches include task-bundle generation, expansion of shared environments, and policy-conditioned curriculum design. WorkRover jointly grounds Tasks and Verifiers in recoverable Work Contexts and versioned Harness capabilities. Its construction pipeline shares execution-based validation with this literature; the repair procedures are compared separately in Appendix A.4.

A.3 Verifier Reliability Under Changing Workspace State

Grounding the acceptance rule. Grounded evaluation requires agreement between instructions and evidence (He et al., 2026a;b). Anchor compiles instructions, environments, solver-certified solutions, and verifiers from a shared workflow specification (Ivanov & Rana, 2026). DeepRubric co-constructs queries and rubrics from evidence trees (Zhu et al., 2026). Universal Verifier fixes criteria before inspecting trajectories and separates process, outcome, and uncontrollable failures (Rosset et al., 2026). AutoSciRub grounds rubrics before execution; RubricRefine uses task/tool rubrics for pre-execution program repair (Wang et al., 2026b; LeVine et al., 2026). RubricEval and RuVerBench test criterion-level judgment accuracy (Pan et al., 2026a; Peng et al., 2026b).

Verifier auditing. EvilGenie, Hack-Verifiable Environments, and RewardHackingAgents expose exploitation, evaluator tampering, and leakage (Gabor et al., 2025; Roth et al., 2026; Atinafu & Cohen, 2026). Adversarial hacker–fixer loops patch tests and environments while checking legitimate solutions (Zhong et al., 2026b).

This literature evaluates both false acceptance and false rejection, including the risk that stronger rejection checks exclude legitimate solutions. WorkRover uses negative mutants and independent valid alternatives under an instruction-grounded Evaluation Contract. The Contract and its criterion–evidence bindings remain fixed during Verifier revision; candidate implementations are compared on shared evidence against independent anchors. Evaluation-invalid episodes are excluded from training. These checks provide evidence within the declared Contract and audit coverage.

A.4 Environment Evolution and Repair Selection

Reuse and adaptation. Voyager, ExpeL, SkillCraft, and ReUseIt study reusable skills and experience (Wang et al., 2023; Zhao et al., 2023; Chen et al., 2026; Liu et al., 2026b; Piriyakulkij et al., 2026). CoEvoSkills, SkillHone, SkillForge, and SkillAdaptor extend reuse with verification, diagnosis histories, and execution-tested revisions (Zhang et al., 2026a; Li & Hu, 2026; Yang et al., 2026b; Yu et al., 2026b). CoEvolve guides task synthesis and training with failure-prone interactions (Yang et al., 2026a). EnvHarness uses rollout diagnoses to customize difficulty and horizon while retaining the original verifier (Huang et al., 2026). Meta-Harness, Agentic Harness Engineering, and StarHarness optimize the agent's execution scaffold from observed outcomes (Lee et al., 2026; Lin et al., 2026a; Esakkiraja et al., 2026).

Reported repair procedures. FACET routes a failed validation to an implicated artifact, then repeats validation from a clean state while retaining the source specification (Shi et al., 2026, §2.4, Algorithm 1). Envs-FORGE selects among six synthesis actions per seed before jointly rewriting and verifying the task bundle (Wu et al., 2026b, §§3.3–3.4). These published procedures organize search around routed repair operations or synthesis actions. FCR freezes component-level proposals before replaying their joint combinations; it also checks regressions in affected tasks.

Search and revision objectives. Patch minimization is established in automated program repair: GenProg searches for test-passing variants and reduces successful patches using structural differencing and delta debugging (Weimer et al., 2009). FCR applies bounded enumeration to proposals over Context, Task, Harness, and Verifier. Each combination is materialized with its dependency closure and replayed under matched conditions; the selection criterion counts direct and closure-induced changes subject to fixed requirements and applicable validation gates. Its minimum is restricted to the tested combinations. Table 4 summarizes the construction and maintenance capabilities under explicit operational criteria.

A.5 Long-Horizon Learning and Progress Attribution

Training interfaces and local feedback. Agent Lightning and RLFactory provide agent-training interfaces (Luo et al., 2025; Chai et al., 2025). ClawGym II and OpenForgeRL support training within native or opaque harnesses (Song et al., 2026a; Yu et al., 2026a). Related work studies long-horizon execution, supervision, and efficiency (Ma et al., 2026; Hao et al., 2026; Cheng et al., 2026a; Xi et al., 2025b; Sun et al., 2025; Liu et al., 2026c).

Methods for local learning signals differ in the evidence used to evaluate decisions (Xi et al., 2025c; Zhao et al., 2024). Beyond GRPO's trajectory advantages, methods use repeated states (GiGPO), action structure (A3, HiPER), hindsight Q-values, and comparable tool decisions (TAPO) (Shao et al., 2024; Feng et al., 2025; Su & Wen, 2026; Peng et al., 2026a; Tan et al., 2026a; Dong et al., 2026a). Learned progress (SPA-RL), entropy (CARL), and rubric or action-role judgments (ARCO,

TRIAGE) offer other local signals (Wang et al., 2025; Shen et al., 2025; Tian et al., 2026; Xu et al., 2026b).

Progress signals. TRACE uses temporal differences in frozen-model reference-answer likelihoods, with a bounded look-ahead window for delayed tool effects (Tao et al., 2026, §3). TRCA combines deterministic transition-rubric scores with increments in cumulative condition coverage. Its coverage component records whether a condition has previously been met; separate local scores include invalid-action penalties (Zhang et al., 2026b, §3, Appendix C). VTCA uses verifier-grounded progress attribution for persistent, mutable workspace artifacts: frozen checkers score successive observable states, and recorded version dependencies determine eligible prerequisite allocations. The signed differences can register both increases and decreases in checked completion. They are progress proxies for advantage shaping, with semantics defined by the checkers and their evidence coverage.

Evidence localization. STAMP maps verified citations to first-exposure actions, Rubric-to-Code Credit Assignment maps rubric judgments to generated code spans, and DRACO uses judge-attributed steps with sign-preserving GRPO advantage redistribution (Xu et al., 2026a; Jin et al., 2026; Gandhi et al., 2026). VTCA's allocation follows recorded direct use of exact tool outputs or artifact versions, sharing at most half of a positive checker increment with earlier eligible turns and leaving negative increments local. Access alone supplies no dependency edge. This rule addresses prerequisite coverage, an issue also examined in analyses of concentrated credit (Zhou et al., 2026a). Provenance quality limits attribution quality: incorrect links can misallocate shaping signal, while missed links leave relevant prerequisites unrepresented. Dependency precision and recall, checker coverage, and localization are audited in Appendix B.8. Table 4 summarizes the local-signal mechanisms; recorded dependencies do not establish causal effects.

Reward shaping and redistribution. Potential-based reward shaping studies transformations that preserve optimal policies under stated assumptions (Ng et al., 1999); RUDDER learns return decompositions for reward redistribution (Arjona-Medina et al., 2019). VTCA is formulated as workspace-grounded advantage shaping. Its conservation properties and the distinction from policy-invariant reward transformations are discussed in Appendix B.8.

A.6 Criteria for the Method Comparison

Table 4: Comparison of reported method capabilities. ✓ denotes full coverage of the stated criterion, ✓ partial coverage, and × no coverage in the reported procedure. Appendix A.6 defines the criteria.

Method	Task + verifier co-design	Component revision	Minimal revision	Cross-task regression	Turn-level shaping	Rule-based progress	Provenance allocation
FACET (Shi et al., 2026)	✓	✓	✓	×	×	×	×
Envs-FORGE (Wu et al., 2026b)	✓	✓	×	×	×	×	×
TRACE (Tao et al., 2026)	×	×	×	×	✓	×	×
TRCA (Zhang et al., 2026b)	×	×	×	×	✓	✓	×
STAMP (Xu et al., 2026a)	×	×	×	×	✓	×	✓
DRACO (Gandhi et al., 2026)	×	×	×	×	✓	×	×
WorkRover	✓	✓	✓	✓	✓	✓	✓

Construction and maintenance criteria. *Task/verifier co-design* constructs executable tasks and their acceptance checks from shared task or environment evidence. *Component revision* updates existing environment, task, harness, or verifier artifacts through repair or evolution. *Minimal revision* uses an explicit revision-size objective to select among validated candidates. FACET receives partial coverage here because it preserves valid components through targeted repair without an explicit minimum-over-candidates selection objective (Shi et al., 2026, §2.4, Algorithm 1). FCR's minimum is restricted to its tested combinations. *Cross-task regression* rechecks other tasks affected by a shared environment change. The repair and synthesis procedures are discussed in Appendix A.4.

Learning-signal criteria. *Turn-level shaping* introduces local scores or allocation weights into policy optimization beyond a uniform trajectory-level advantage. *Rule-based progress* evaluates intermediate task conditions with deterministic state or transition checks; terminal-only tests and model-scored answer likelihoods or rubric judgments do not meet this criterion. TRCA's transition rubrics satisfy it, as do VTCA's supported workspace checkers. *Provenance allocation* selects signal recipients through explicit source or data-use links. Both STAMP's citation-to-source attribution and

VTCA's recorded artifact-version dependencies satisfy this criterion; their different evidence units are described in Appendix A.5.

Scope of the entries. Symbols refer to the published procedures cited in the table and the operational criteria above. The WORKROVER row describes the complete framework: construction and FCR supply the maintenance capabilities, while VTCA supplies the local learning signals. A check denotes a mechanism's presence under its stated scope, including checker and provenance coverage limits; it does not imply a performance or causal-attribution guarantee.

B ADDITIONAL METHOD DETAILS

B.1 WORK CONTEXT CONSTRUCTION DETAILS

File indexing and data splits. Let $\mathcal{W}$ be the curated set of source workspaces. For each $w \in \mathcal{W}$, WORKROVER builds a typed file-relation graph

$$G_w = (V_w, E_w), \qquad E_w = E_w^{\text{sem}} \cup E_w^{\text{str}} \cup E_w^{\text{tmp}} \cup E_w^{\text{prov}} \cup E_w^{\text{flow}}. \tag{2}$$

Nodes denote files or file versions, and edges encode semantic similarity, structural references, temporal order, provenance, and workflow dependencies. Directory co-location is retained only as a weak structural signal. The index $I(w)$ stores this graph with file type, editability, and lineage metadata. Workspaces linked by recorded lineage, together with their derived task branches, are assigned to the same data split.

Skill matching and environment assembly. For job skill j, the requirement schema $R(j)$ records its professional setting, workflow stages, required materials, and expected deliverables. Once j is selected, the Skill Matcher retrieves k candidate file regions, each comprising related nodes in a source workspace's graph:

$$\mathcal{R}_j = \text{TopK}(\text{Retrieve}\,(R(j), \{I(w) : w \in \mathcal{W}\})). \tag{3}$$

Each retrieved region serves as a starting point for expansion through file and application skills with compatible inputs, outputs, and interfaces. Application instrumentation provides runtime controls, UI observations, interface semantics, reset and state-capture operations, and hooks for collecting state evidence. Joint assembly binds these capabilities to the selected files under the applicable Harness permissions and recording requirements. Before retaining a context, the Harness checks that required dependencies are accessible, bindings and permissions agree with H, and reset and recovery restore the declared observable state.

Construction cost. Let J be the number of job skills, $W = |\mathcal{W}|$, k the number of shortlisted regions per job skill, and $M \leq Jk$ the number of candidates retained for environment assembly. Let C_{index} and C_{retrieve} denote per-workspace indexing and per-query retrieval costs; C_{agent}, C_{match}, and C_{env} denote per-candidate agent inspection, skill matching, and environment-assembly costs. Excluding source collection and one-time pool construction, the total cost of indexed construction is

$$O(WC_{\text{index}}) + O(JC_{\text{retrieve}}) + O(Jk(C_{\text{agent}} + C_{\text{match}})) + O(MC_{\text{env}}). \tag{4}$$

The agent-inspection term scales with the Jk shortlisted candidates; exhaustive job–workspace inspection instead incurs $O(JWC_{\text{agent}})$. The other terms account for indexing, retrieval, skill matching, and environment assembly.

B.1.1 JOB ROLE INVENTORY

The job-skill library covers 317 unique role identifiers, listed below with English names and grouped into 12 source categories. These groups organize individual and team roles; they are distinct from the 10 secondary skill categories in Figure 14.

Product & Design (16 roles). 1. Ardot Design Expert; 2. AI Image Prompt Engineer; 3. Behavioral Nudge Engine; 4. Delightful Experience Designer; 5. Design Prototyping Expert Team; 6. Design System Architect; 7. Design-to-Code Expert; 8. Feedback Synthesis Analyst; 9. Diagram Design and Rendering Expert; 10. Product Management Expert; 11. Product Strategy Team; 12. Sprint Priority Manager; 13. UI Designer; 14. User Experience Architect; 15. User Experience Researcher; 16. Visual Storytelling Expert.

Engineering (42 roles). 17. Android Engineer; 18. CAD Engineer; 19. C/C++ Engineer; 20. Electrical Engineer; 21. Environmental Engineer; 22. Go Engineer; 23. iOS Engineer; 24. Java Engineer; 25. Mechanical Engineer; 26. Network Engineer; 27. PHP Engineer; 28. Python Engineer; 29. Rust Engineer; 30. AI Coding Architecture Expert Team; 31. Backend Architect; 32. Tencent Cloud Technical Support; 33. Code Review Expert; 34. DevOps Automation Engineer; 35. Dockerfile Generation Expert; 36. Embedded Firmware Engineer; 37. Engineering Assurance Team; 38. Engineering Practice Expert; 39. Frontend Developer; 40. Git Workflow Expert; 41. Incident Response Commander; 42. Infrastructure Operations Expert; 43. LSP

Index Engineer; 44. MCP Build Expert; 45. Tencent Cloud Migration Expert Team; 46. Mobile Application Developer; 47. Modern Web Development Expert; 48. Rapid Prototyping Engineer; 49. Security Engineer; 50. Senior Developer; 51. Site Reliability Engineer; 52. Software Architect; 53. Software Development Team; 54. Software Workshop; 55. Solidity Smart Contract Engineer; 56. Terminal Integration Expert; 57. Threat Detection Engineer; 58. WeChat Mini Program Developer.

Games & Spatial (22 roles). 59. Game Audio Engineer; 60. Game Designer; 61. Godot Game Script Engineer; 62. Godot Multiplayer Engineer; 63. Godot Shader Developer; 64. Level Designer; 65. Narrative Designer; 66. Roblox Avatar Creator; 67. Roblox Experience Designer; 68. Roblox System Script Engineer; 69. Technical Artist; 70. Unity Architect; 71. Unity Editor Tool Developer; 72. Unity Multiplayer Engineer; 73. Unity Shader Graph Artist; 74. Unreal Multiplayer Architect; 75. Unreal System Engineer; 76. Unreal Technical Artist; 77. Unreal World Builder; 78. visionOS Spatial Engineer; 79. XR Immersive Developer; 80. XR Interface Architect.

Data & AI (29 roles). 81. Academic Research Assistant; 82. BI Engineer; 83. Bioinformatics Engineer; 84. Business Analyst; 85. Database Administrator; 86. GIS Analyst; 87. Healthcare Data Analyst; 88. Machine Learning Engineer; 89. Prompt Engineer; 90. Academic Journal Selection Advisory Team v3.0; 91. Data Analytics Expert Team; 92. AI Engineer; 93. News Digest Expert; 94. Autonomous Optimization Architect; 95. Data Analysis Expert; 96. Data Analytics Reporter; 97. Data Engineer; 98. Data Exploration Expert; 99. Data Integration Agent; 100. Database Optimization Expert; 101. Deep Research Expert; 102. Financial Data Retrieval Expert; 103. Huashu Data Analysis Expert Team; 104. Data Table Creation Expert; 105. Knowledge Management Expert; 106. Model Quality Assurance Expert; 107. Sales Data Extraction Agent; 108. Industry Trend Expert; 109. Intelligent Vocabulary Coach.

Marketing & Growth (35 roles). 110. Amazon Japan Product Data Operations Specialist; 111. Market Researcher; 112. Ad Creative Strategist; 113. Ad Tracking Technology Expert; 114. App Store Optimization Expert; 115. Automotive Marketing Expert; 116. Baidu SEO Expert; 117. Brand Strategist; 118. Carousel Content Growth Expert; 119. China E-commerce Operations Expert; 120. Content Monetization Expert Team; 121. Cross-Border E-commerce Expert; 122. Cultural Intelligence Strategist; 123. Developer Evangelist; 124. Egypt Marketing Expert; 125. Brand GEO Visibility Diagnostician; 126. Growth Hacker; 127. Identity Graph Operator; 128. Advertising Campaign Management Expert; 129. LinkedIn Content Creator; 130. Livestream E-commerce Coach; 131. Marketing Campaign Planning Expert; 132. Marketing Campaign Team; 133. Paid Media Auditor; 134. PPC Bidding Strategist; 135. Private-Domain Marketing Expert; 136. Private-Domain Operations Expert; 137. Programmatic Ad Buyer; 138. Reddit Community Builder; 139. Search Term Analyst; 140. SEO Content Marketing Team; 141. SEO Expert; 142. Social Ad Strategist; 143. Social Media Engagement and Growth Expert Team; 144. Social Media Strategist.

Content & Creative (41 roles). 145. Animator; 146. Copywriter; 147. Content Editor; 148. Grant Writer; 149. Illustrator; 150. Journalist / Data Journalist; 151. Photographer; 152. Podcast Producer; 153. Screenwriter; 154. Translation and Localization Specialist; 155. AI Shifu Course Production Expert; 156. Content Creation Expert Team; 157. Bilibili Content Strategist; 158. Book Co-Creator; 159. Automotive Content Creation Expert Team; 160. Content Creation Expert; 161. Omnichannel Content Distribution Expert Team; 162. Douyin Strategist; 163. Instagram Operations Expert; 164. Document Butler; 165. Knowledge Collection Expert; 166. PDF Processing Expert; 167. AI Presentation Expert; 168. Knowledge Video Creation Expert; 169. Kuaishou Strategist; 170. Long-Document Writing and Revision Expert; 171. News Advisor; 172. Podcast Strategist; 173. Presentation Creation Expert; 174. Promotional Video Creation Team; 175. Video Generation Expert; 176. Short-Video Editing Coach; 177. TikTok Strategist; 178. Twitter Operations Expert; 179. Video Analysis Expert Team; 180. AI Video Creation Team; 181. Viral Topic Planning Expert; 182. WeChat Official Account Operations Expert; 183. Weibo Strategist; 184. Xiaohongshu Operations Expert; 185. Zhihu Strategist.

Sales & Commerce (16 roles). 186. Account Manager; 187. Customer Success Manager; 188. Revenue Operations Specialist; 189. Sales Development Representative (SDR/BDR); 190. Tender Sales and Sourcing Specialist; 191. Deal Strategist; 192. Discovery Coach; 193. Enterprise Account Strategist; 194. Meituan Lifestyle Assistant; 195. Outbound Strategist; 196. Presales Technical Consultant; 197. Proposal Strategist; 198. Sales Execution Team; 199. Sales Coach; 200. Sales Pipeline Analyst; 201. Revenue Growth Specialist.

Finance & Investment (30 roles). 202. Expense Reimbursement Operations Specialist; 203. Internal Auditor; 204. Tax Specialist; 205. A-Share Research Team; 206. Accounts Payable Agent; 207. Citongshuo Pro Investment Community Guest Team; 208. Earnings Researcher; 209. Equity Research Expert; 210. Finance and Accounting Expert; 211. Financial Modeling Expert; 212. Financial Tracker; 213. Industry Researcher; 214. General Ledger Reconciler; 215. Investment Banking Expert; 216. Investment Masters Expert Team; 217. Customer Compliance Officer; 218. Meeting Preparation Assistant; 219. Financial Model Builder; 220. Month-End Closing Accountant; 221. Private Equity Expert; 222. Multi-Market Stock Analyst; 223. Finance Manager; 224. LP Statement Auditor; 225. Quantitative Strategy Backtesting Specialist; 226. Individual Stock Diagnostician; 227. Intelligent Stock Selector; 228. Value Assessor; 229. Trading Analysis Team; 230. Portfolio Valuation Supervisor; 231. Wealth Management Expert.

Operations & HR (16 roles). 232. Administrative Assistant; 233. Compensation and Benefits Specialist; 234. Executive Assistant; 235. Inventory Management Specialist; 236. Office Manager; 237. Procurement

and Supplier Sourcing Specialist; 238. Property Management Specialist; 239. Corporate Training Designer; 240. Customer Support Expert; 241. HR Operations Team; 242. AI Interviewer; 243. Malaysia HR and Administrative Compliance Advisor; 244. Recruitment Expert; 245. Organizational Operations Specialist; 246. Study Abroad Consultant; 247. Supply Chain Strategist.

Project & Quality (23 roles). 248. Accessibility Auditor; 249. Agent Orchestrator; 250. API Testing Expert; 251. Document Generation Expert; 252. Document Processing Expert; 253. Evidence Collector; 254. Strategic Report Advisor; 255. Experiment Tracking Manager; 256. Internal Communications Expert; 257. Jira Workflow Administrator; 258. Professional Document Generation Team; 259. Performance Testing Expert; 260. Project Shepherd; 261. Reality Checker; 262. Report Distribution Agent; 263. Senior Project Manager; 264. Studio Operations Manager; 265. Studio Producer; 266. Technical Documentation Engineer; 267. Test Results Analyst; 268. Tool Evaluation Expert; 269. Web Application Testing Expert; 270. Workflow Optimization Expert.

Security, Legal & Compliance (25 roles). 271. Electronic Contract Signing Operations Specialist; 272. Paralegal; 273. Patent / IP Specialist; 274. Privacy Officer / DPO; 275. Agent Identity and Trust Expert; 276. Blockchain Security Auditor; 277. Chinese Legal Advisory Team; 278. Compliance Auditor; 279. Contract Risk Advisor; 280. Senior Contract Legal Expert; 281. Corporate Legal Expert Team; 282. Board Secretary Assistant; 283. Healthcare Marketing Compliance Expert; 284. Contract and Official Document Reviewer; 285. Intelligent Invoice Expert Team; 286. Law Study Coach; 287. Legal Skills Governance Expert; 288. Legal Compliance Reviewer; 289. Legal Research Expert; 290. Litigation Support Advisor; 291. Marketing Copy Reviewer; 292. Customer and Compliance Officer; 293. Finance and Tax Compliance Expert Team; 294. Litigation Assistant; 295. Zero-Knowledge Proof Administrator.

Industry Consulting (22 roles). 296. Clinical Researcher; 297. Construction Project Business and Risk Controller; 298. Curriculum Designer; 299. Government Affairs Specialist; 300. Instructional Designer; 301. Librarian / Knowledge Librarian; 302. Nonprofit Operations Specialist; 303. Public Policy Analyst; 304. Real Estate Analyst; 305. Brazil Company Information Analyst; 306. Socratic AI Thinking Partner; 307. Entrepreneurship Partner; 308. Family Education Expert; 309. Fubangshou Super Partner; 310. Industry Scenario Researcher; 311. Digital Fortune-Telling Consultant; 312. Gaokao Advisor; 313. Independent Entrepreneurship Coach; 314. Entrepreneurial Mindset Companion; 315. One-Person Company Expert Team; 316. Business Operations Manager; 317. Travel Planning Assistant.

B.1.2 WORKSPACE CONSTRUCTION FROM DEEP RESEARCH SESSIONS

Role in the construction pipeline. The Deep Research sessions in Figure 3 contribute exported file collections to the filesystem pool. In this route, a source workspace w denotes the retained collection associated with a research record. Collections may contain source documents, code, tables, and independently retained notes or reports. The following protocol specifies which materials can support new professional tasks and how the resulting candidates enter the standard admission process. Historical research records are distinct from the Demonstrator and training rollouts collected for the new tasks (Appendix B.7). Figure 8 summarizes this construction route and its downstream WORKROVER admission. Recorded examples of this source material are presented in Appendix B.1.3.

File inventory and eligibility. Each exported collection is ingested under its source workspace identity. Available files are inventoried by relative path, format, size, and content hash. Deterministic inspection produces bounded previews and format-specific structure, such as document headings, workbook sheets, and archive members. This shared index supports matching and selective reading without repeatedly processing the full collection. Raw traces, internal analysis caches, provenance directories, and unsupported web-cache files are excluded from the selectable input pool. The original research query is retained only as background for task construction. It may inform job skill selection when supported by the available files, but is excluded from Solver inputs and cannot serve as task evidence. Table 5 summarizes these material roles and visibility rules.

Source identity and file relationships. Available source references and derivation records supplement the file content when constructing G_w. Structural references and shared entities help identify related documents; observed versions and transformations can support temporal or provenance edges. A citation in a generated note suggests a link to the cited document; the document must be checked before that link is treated as evidentiary support. Co-occurrence in a conversation, retrieval order, or an unsupported model assertion does not establish a factual dependency. Missing lineage is recorded as unresolved.

Freezing local files fixes the task's input state; it does not imply that the original searches, web pages, or remote services can be replayed. A candidate that depends on an unavailable source must be revised to an objective supported by the available evidence or rejected. Any permitted external

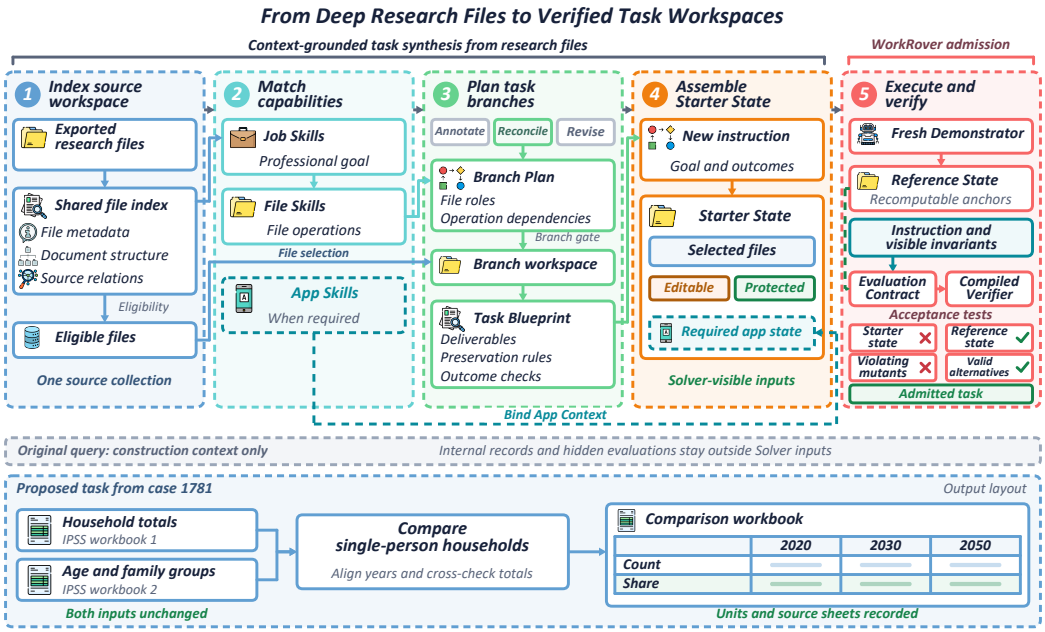

Figure 8: **From Deep Research files to verified task workspaces.** Indexed files support task construction, fresh execution, and verification. Instructions and visible invariants define requirements; the dashed green arrow supplies recomputable Reference State anchors. The proposed case 1781 comparison uses two IPSS workbooks; shaded cells mark output values to populate.

Table 5: Roles of materials associated with a Deep Research session. Eligibility for selection does not make a file authoritative or sufficient for a new task.

Material	Construction role	Solver visibility
Captured source files	Candidate evidence, assessed for readability, task relevance, and available source information.	Selected, frozen files may enter the Starter State.
Notes, tables, and reports	Independently retained, readable documents for comparison or editing; factual claims require supporting evidence.	Eligible only with a declared role and without supplying the new task's completed answer.
Original research query	Background for interpreting the collection and adapting task intent.	Retained for construction; excluded from Solver inputs.
Conversation, tool logs, and internal analysis	Collection context and audit records; any usable source reference must resolve to admissible evidence.	Excluded from the selectable input pool.
Provenance and source pointers	Lineage and source identity metadata; a URL cannot replace a required local source.	Internal provenance manifests are excluded from Solver inputs.

retrieval follows the declared Harness contract and records the newly obtained evidence separately from the historical collection.

Task discovery and branch construction. The indexed collection and the original query as construction context support selection of a compatible job skill j. Its requirement schema $R(j)$ then guides retrieval of eligible file regions within that collection, following the indexed retrieval formulation above. The Skill Matcher matches file skills to the selected file roles and operation requirements. When a branch requires application operations, compatible application skills are included in the Capability Context and bound through the App Context to instrumented environments under the declared Harness. The Matcher may adapt the original research topic into a bounded task, such as reconciling tables, annotating a manuscript, checking conflicting claims, or revising an existing report. Each task branch selects files from one source collection w, and those files must support its objective. Branches must differ in the professional decisions they require, the operations they involve, or their principal work products. A branch may contain multiple source documents, but task authoring cannot import unselected files or combine different source collections. Each Branch Plan is

checked for feasibility and distinctiveness, then materialized as a workspace containing the selected files and cached relationships. The Task Blueprint refines this plan into a structured specification of file roles, operation dependencies, deliverables, and modification and preservation constraints. Each required outcome is linked to an observable evaluation criterion.

Historical outputs and answer leakage. A historical note or report is eligible for editing or comparison only when it exists as an independently retained, readable file. A logged write acknowledgment or an excerpt returned by a tool alone does not supply an editable Starter State artifact. Claims in retained documents remain subject to the same evidence requirements as other working materials. A candidate whose visible inputs already satisfy its required outcomes must be revised or discarded. Internal plans, hidden evaluation records, and historical answer text used only as a solution key remain outside the Solver-visible state. Every fact needed to complete the new task must be supported by visible materials or an explicitly declared external capability. Hidden provenance may support an audit, but cannot introduce unstated requirements or substitute for evidence accessible to the Solver.

Example: household-projection comparison. Consider the two IPSS household-projection workbooks described in Figure 10. They belong to the exported file collection associated with case 1781. A proposed task asks the Solver to compare single-person household counts and shares for 2020, 2030, and 2050. Its Task Blueprint connects category identification, year alignment, value extraction, cross-workbook consistency checks, and unit checks while permitting alternative execution paths. The deliverable would be a new comparison workbook with category definitions, units, and source-sheet references; both input workbooks remain unchanged. This example illustrates a new task supported by selected source tables, rather than a reconstruction of the historical policy memorandum.

Fresh execution and verification. The selected artifacts and any required application state define a frozen Starter State for the new instruction. A separate Demonstrator executes the instruction from the frozen Starter State to establish feasibility and produce a Reference State. Completion of the historical research session cannot substitute for this execution. The Evaluation Contract derives its requirements from the instruction and declared agent-visible invariants; evaluation anchors must be independently recomputable, as specified in Appendix B.3. Counterfactual validation checks that the Verifier rejects the Starter State and accepts a valid Reference State and valid alternative solutions. It also tests rejection of mutants that violate required outcomes or protected-state constraints. Candidates must pass the standard admission protocol; file inventory and package-format checks alone are insufficient.

Lineage and accounting. Branches retain their source collection identity and available research-session and file-version lineage, and follow the data-split rule above. Research sessions, retained collections, task candidates, and admitted tasks are counted separately. Neither session completion nor branch count establishes independent task diversity. Source availability, file eligibility, evidence sufficiency, answer leakage, and downstream admission remain distinct rejection stages.

B.1.3 Recorded Deep Research Examples

Figures 9 and 10 pair recorded research workflows with their exported file collections. m_i is the zero-based trace message index. The proposed file tasks illustrate Appendix B.1.2. Shaded cells indicate output fields to populate.

Source-page references associate these files with the research but do not establish per-file downloads in the recorded traces. New tasks still require fresh Demonstrator execution and counterfactual Verifier checks; raw traces and internal provenance remain outside Solver inputs.

B.2 Task Synthesis and Quality Control

Task package and workspace coherence. Each instance contains an instruction, a resettable workspace snapshot, an environment specification, structured metadata, and, when available, a Demonstrator realization with terminal artifacts. Instructions state the professional objective, material categories, deliverables, and outcome constraints without exposing paths, exact filenames, tools, latent skills, or action sequences. Core artifacts must have semantic, temporal, provenance, structural-reference, or workflow relations; directory co-location alone does not establish relevance. Input ablation removes bindings that are unnecessary for every outcome criterion, while distractors may increase navigation difficulty without changing the dependency closure.

Difficulty calibration. We record workflow horizon, state dependence, artifact number and diversity, cross-root dependency depth, exact and negative constraints, contextual noise, expected state

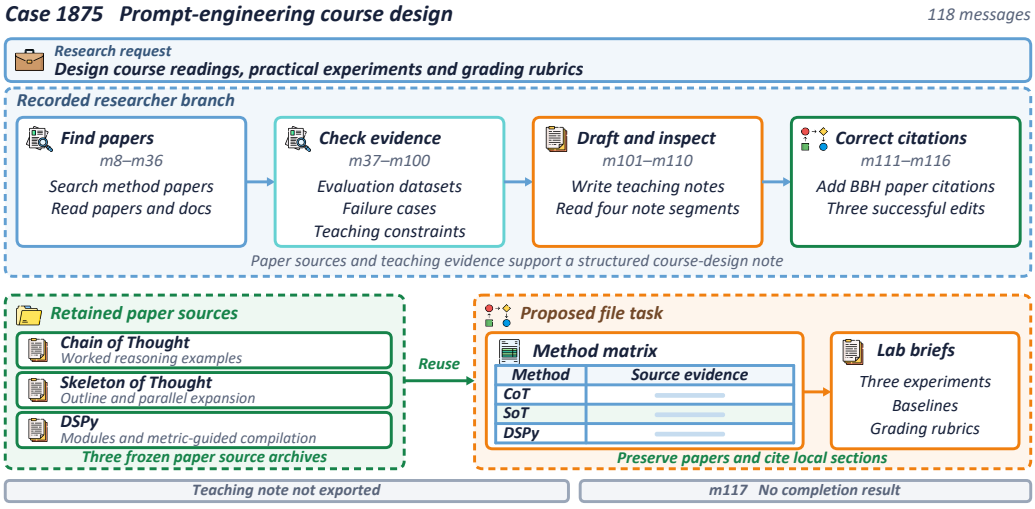

Figure 9: **Recorded example 1875: prompt-engineering course design.** This researcher branch writes `teaching_design.md`, inspects four segments, and adds the BIG-Bench Hard paper citation in three places. The proposed task reuses captured CoT, SoT, and DSPy paper-source archives to produce a method comparison workbook and three lab briefs with baselines and grading rubrics. Claims cite local source sections; the original archives remain unchanged.

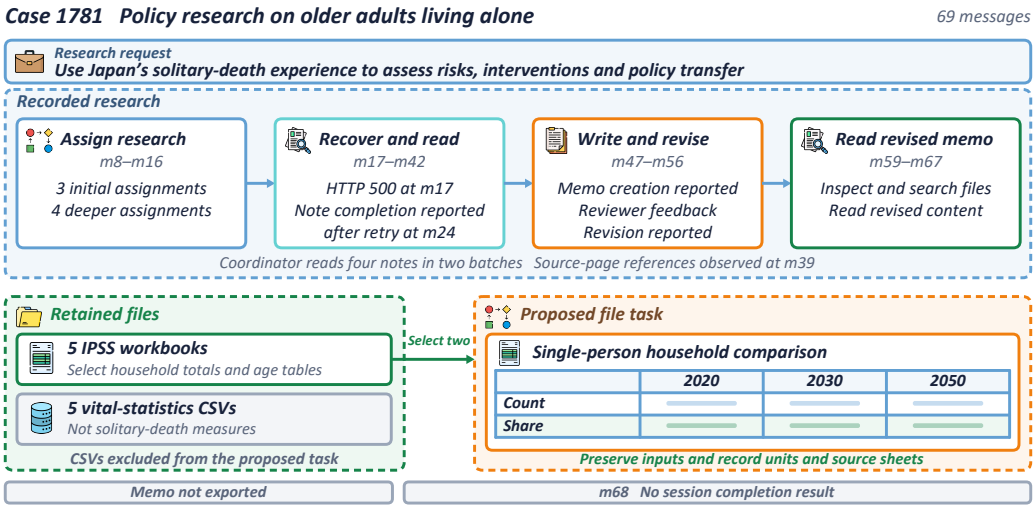

Figure 10: **Recorded example 1781: policy research on older adults living alone.** The proposed task selects `hprj2024_gaiyo_kekka1.xlsx` and `hprj2024_gaiyo_kekka2.xlsx`, retaining category definitions, units, and source-sheet references. The revised `policy_memo.md` is observed through tool readbacks. Source-page references at m_{39} cite IPSS and MHLW.

Table 6: Target task-construction profiles. The profiles control scale and interaction horizon but do not define empirical difficulty.

Dimension	P1	P2	P3
Edit scope (files)	∼10	∼40	∼80
Demonstrator horizon target (turns)	5–10	15–30	30–50
Skill composition target	2 job + $\geq$ 1 file	3 job + $\geq$ 3 file	4 job + $\geq$ 5 file
App skills	As required	As required	As required
WorkEnv noise ratio	10%	20%	30%
Cross-env dependencies	2	3	4
Distractor tiers	Obvious	Obvious + near-topic	All three

changes, and recovery requirements as construction descriptors, not as difficulty labels. Empirical difficulty is an operational, policy-relative quantity: for each validated task, a frozen reference solver panel performs repeated rollouts from an identical snapshot under the same scaffold, tools, permissions, and budget. Difficulty is one minus the aggregate valid-rollout success rate, equivalently the valid-rollout failure rate, and determines the calibrated band. Panel membership, aggregation, band thresholds, and k are fixed on a held-out development split before evaluation; evaluated systems never contribute to the labels. We report the profile and calibrated band separately, together with outcome score, criterion completion, pass@1, pass@k, turns, tool errors, and cost with bootstrap confidence intervals. Turns and cost diagnose effort but never determine the band, and we report their association with calibrated difficulty rather than assuming that longer trajectories are harder.

Verifier readiness. Every essential requirement maps to a lexical, numerical, structural, artifact, or relational anchor. Aggregations specify their window, filters, grouping rules, and missing-value treatment; file-producing tasks specify observable artifact properties. Subjective goals are decomposed into enumerable minimum conditions, and candidates with unobservable core outcomes are removed. The Demonstrator trace, terminal diff, and artifacts establish feasibility and ground the rubric while permitting alternative valid solutions. Executable checks require that the Starter State fails, the Reference State passes, removing a mandatory output causes failure, and protected-state changes are detected. Contract predicates are resolved against canonical file, database, application-object, rendering, or cross-artifact adapters before a task is retained. Evaluation distinguishes pass, fail, and infrastructure-level invalid; only valid episodes enter task-performance denominators or training updates, while invalid-attempt rates are reported separately. Negative mutants cover local attributes, structure, cross-artifact consistency, protected state, semantic near misses, and adversarial artifacts. Alternative valid states test whether the same Verifier admits different formulas, layouts, or execution paths with equivalent outcomes.

Capability contract and reproducibility. Tasks are workspace-local unless an external capability is explicitly declared. Web retrieval, visual interpretation, and other optional capabilities are exposed through Harness tools rather than assumed as native model functions. Environment versions, seeds, hashes, permissions, and initial-state identities are recorded, and repeated rollouts restore the same initial workspace state.

Quality control and diversity. Automated checks cover package structure, file readability, metadata completeness, path privacy, output-map integrity, and reproducible restoration. Semantic review covers instruction clarity, contextual sufficiency, execution consistency, cross-file necessity, task openness, and outcome verifiability. Stratified human review measures professional realism, internal coherence, criterion–evidence agreement, reviewer agreement, and uncertainty. Dataset statistics report roles, workflow families, artifact and output types, prompt structures, capabilities, and calibrated difficulty to detect template collapse.

B.2.1 Task Design and Review

Task design links a professional objective to the available Work Context and to observable outcomes. The Task Blueprint records required inputs, workflow dependencies, deliverables, and modification and preservation constraints. Review checks that each required outcome has evidence that can be extracted from an artifact or application state, or independently recomputed. Table 7 gives representative workflow examples; the criteria for each instance follow its instruction and declared agent-visible invariants.

Workspace construction preserves the artifact features needed to carry out and assess these workflows, including formulas, merged cells, annotations, version history, layout metadata, and embedded objects. The manifest distinguishes required inputs, contextual history, distractors, and protected artifacts. Review checks that required inputs can be opened or parsed and that their task-relevant structures remain intact in the Starter State. When completion depends on visual properties, review also inspects the rendered artifacts.

B.2.2 Recorded Task Example: Quarterly Executive Handoff

We illustrate a quarterly executive handoff task with record `0805:comp_018554` from the 5,000-trajectory pool. This record also belongs to the 3,200-trajectory subset in Figure 5. Figure 11 follows its Hermes execution and shows how evidence from application state supports file updates and the requested deliverables.

Table 7: Representative workflow patterns and task-grounded outcome evidence. Each instance derives its criteria from its own instructions.

Workflow	Context and dependencies	Observable outcomes
Artifact revision and governance	Source artifacts, revisions, review comments, and approval records.	Required edits, consistent versions, retained review history, and unchanged protected content.
Data analysis and reconciliation	Related datasets, formulas, source records, and reporting periods.	Recomputable aggregates, correct formulas, consistent units, and declared treatment of missing values.
Structured content production	Existing reports, templates, charts, media, and source materials.	Required content and organization, specified layout properties, and preserved unaffected elements.
Cross-artifact delivery	Linked artifacts, output specifications, and exported versions.	Complete outputs, correct formats and ordering, and consistency between source and exported content.
Collaboration and coordination	Calendar, email, shared-drive, and approval state linked to project artifacts.	Required application-state changes, consistent records, and evidence that declared workflow conditions hold.

Task dependencies. The revenue figures initially await source verification and growth calculations, the vendor is marked as absent from the ledger, and the five expected exports are unverified. The agent checks the revenue against five regional source rows, registers the vendor, amends the Q2 access policy, and retires the archived Q1 policy before completing the working files. The original revenue values match the source; the completed workbook adds totals, growth calculations, and evidence references. The final report retains the database lock wait and the incident record's stale export alias as follow-up findings.

Recorded evaluation. The archived evaluator marks all four goal groups as passed and assigns a score of 0.946. Recorded file validation covers structure and content; visual review was unavailable in that execution.

B.3 Full Verifier Specification and Calibration Protocol

This subsection records the operational details omitted from Section 3.1.3 to keep the main presentation focused on the Contract–Verifier–decision–calibration chain.

Contract proposal and complete schema. Given Task T_i, Starter State x_i^0, and workflow bindings Π_i, the proposal model emits a criterion skeleton before consulting the Demonstrator Reference State:

$$\mathcal{Q}_i^{(0)} = \mathrm{Propose}_{\theta_Q}(T_i, x_i^0, \Pi_i). \tag{5}$$

The Contract is hidden from the evaluated policy. Every proposed predicate must be traceable to an instruction-level outcome or a declared agent-visible workspace invariant. After the skeleton is fixed, $x_i^\star$ may instantiate an independently recomputable acceptance anchor and Π_i may resolve a dependency or evidence surface, but neither may add a requirement or prescribe the Demonstrator's action sequence. The compiler then grounds the proposal into

$$\mathcal{Q}_i = \left(O_i, \Omega_i^{\mathrm{edit}}, \Omega_i^{\mathrm{protect}}, \Delta_i^{\mathrm{must}}, \Delta_i^{\mathrm{allow}}, \Delta_i^{\mathrm{forbid}}, \{q_{i,k}\}_{k=1}^{K_i}\right). \tag{6}$$

O_i is the required-outcome inventory, Ω_i^{edit} and $\Omega_i^{\mathrm{protect}}$ are the editable and protected state domains, and the three Δ_i fields contain required, allowed, and forbidden typed transition predicates, respectively. The proposal is not treated as ground truth: unresolvable objects, unobservable criteria, and anchors inconsistent with the Starter–Reference contrast are rejected. If a rejected field expresses a core task outcome, the Task Candidate is routed to candidate-stage FCR when the mismatch can be repaired while preserving its frozen criterion skeleton; otherwise it is discarded rather than weakened by removing that field.

Contract fields and grounding. For criterion $q_{i,k} = (d_{i,k}, \zeta_{i,k}, \mathcal{S}_{i,k}, m_{i,k}, w_{i,k}, \gamma_{i,k}, b_{i,k})$, $d_{i,k}$ is a self-contained check, $\zeta_{i,k}$ is its acceptance anchor, $\mathcal{S}_{i,k}$ lists its admissible evidence surfaces, and $m_{i,k} \in \{\mathsf{det}, \mathsf{sem}\}$ selects an executable or semantic decision path. The remaining fields satisfy $w_{i,k} > 0$, $\gamma_{i,k} \in (0, 1]$, and $b_{i,k} \in \{0, 1\}$ and give its weight, threshold, and criticality. Surfaces such as `mcp`, `skill`, `file`, `web`, `coding`, and `integration` specify where evidence must be captured rather than how success is decided. The compiler resolves Contract predicates against canonical file, database, application-object, rendering, and cross-artifact adapters. It rejects

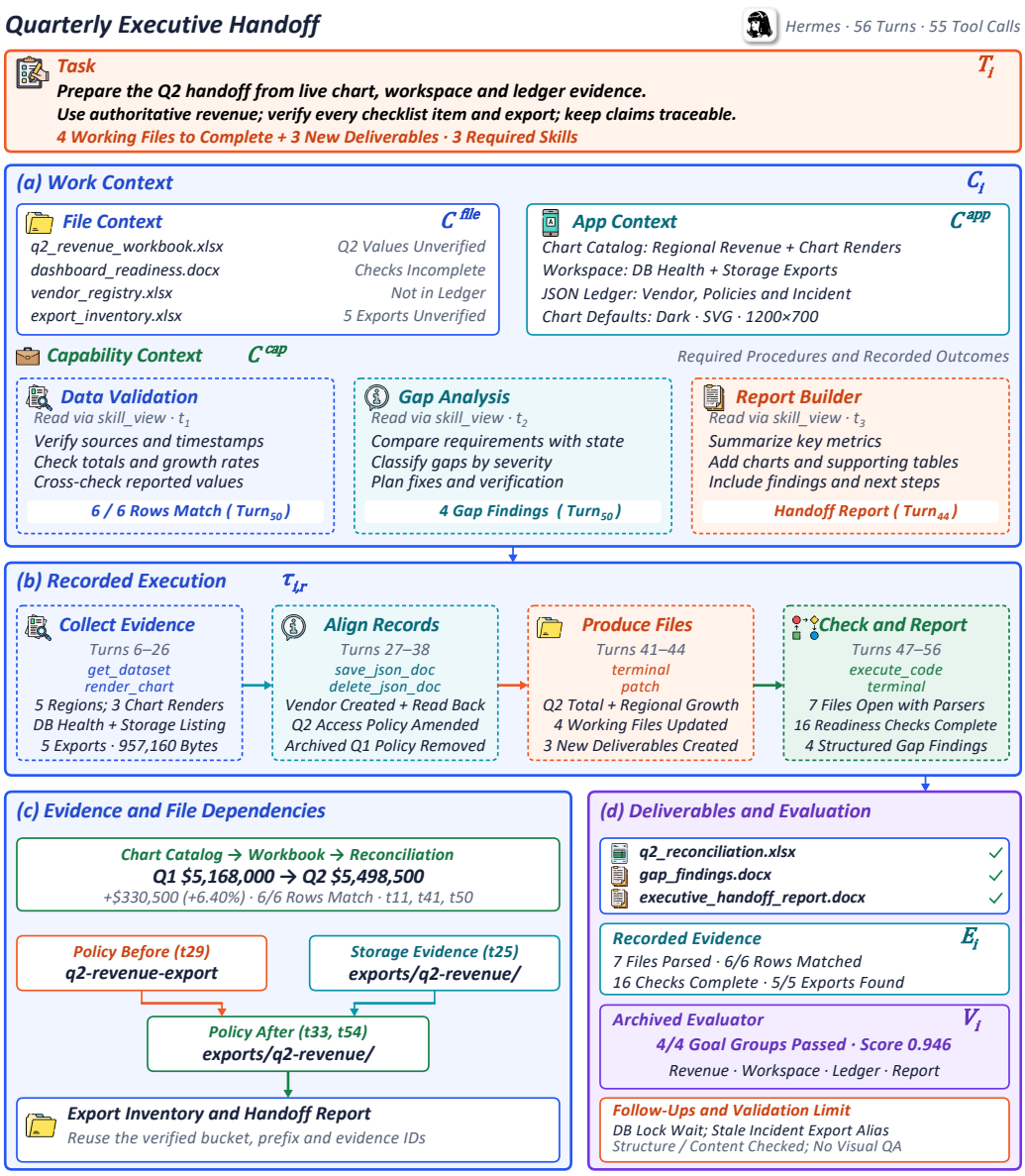

Figure 11: Recorded quarterly executive handoff task T_i. (a) Work Context C_i comprises the working files C^{file}, required skill procedures C^{cap}, and applications C^{app}. (b) Selected stages of rollout $\tau_{i,r}$, with abbreviated tool names. (c) Revenue and export-path evidence support consistent file updates. (d) Deliverables, recorded evidence $\mathcal{E}_i$, and archived evaluator V_i, including retained follow-ups and the scope of file validation.

unresolved roots, unexecutable predicates, criteria without an admissible evidence source, and anchors that disagree with the Starter–Reference contrast. For VTCA, an optional progress-check contract additionally records the frozen checker and schema hashes, supported state fields, evaluability conditions, score range, and admissible provenance links. Development controls include partial completion, verified absence, missing observations, regressions, and alternative valid states. A criterion receives local credit only if this contract passes calibration; otherwise its terminal evaluation is retained. Checks must accept equivalent valid states and cannot reward conformity to the Demonstrator's action sequence or incidental artifact layout.

Canonicalization and artifact-native checks. The canonical diff records typed atomic changes rather than raw byte patches:

$$\widehat{\Delta x}_i = \mathrm{Canon}(x_i^0, x_i^{\mathrm{term}}). \tag{7}$$

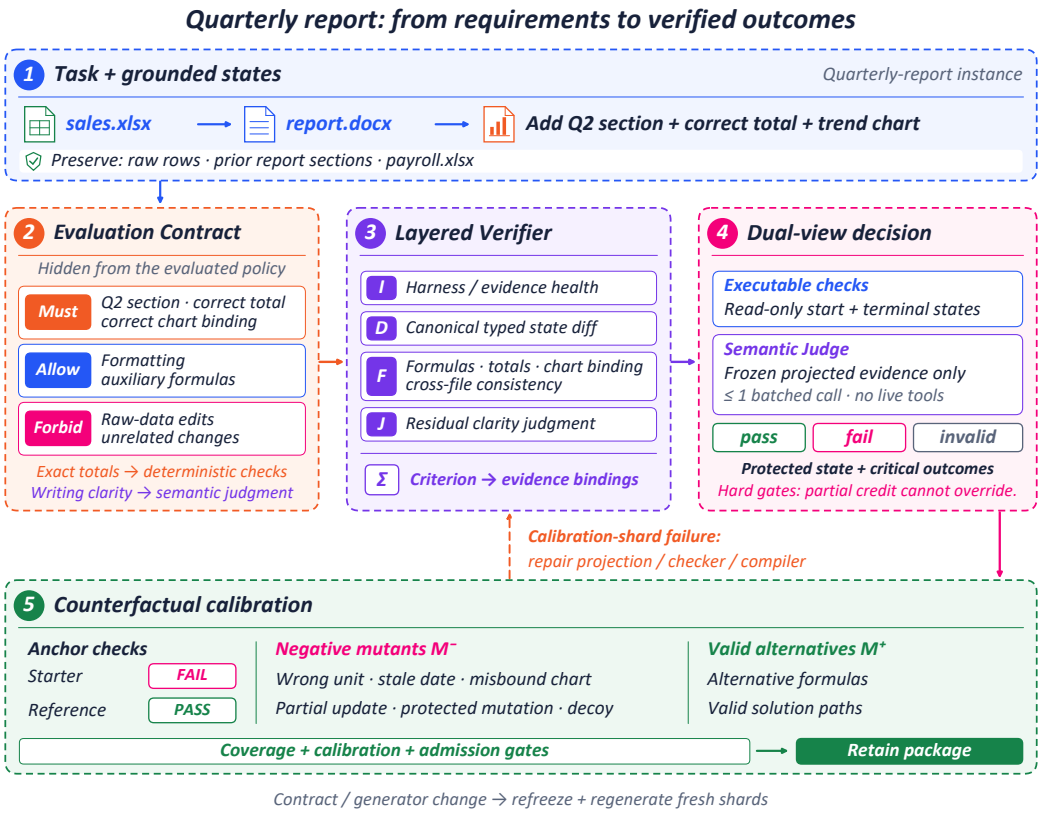

Figure 12: A quarterly-report task traversing the complete WORKROVER Verifier pipeline. The same grounded requirements follow the task from contract proposal through typed state checks, scoped semantic judgment, hard-gated three-valued scoring, and positive/negative counterfactual calibration. The dashed loop denotes localized repair rather than complete task regeneration.

The Contract is enforced through typed matching and admissibility:

$$\bigwedge_{\delta \in \Delta_i^{\text{must}}} \text{match}(\delta, \widehat{\Delta x}_i) = 1, \quad \bigwedge_{\delta \in \Delta_i^{\text{forbid}}} \text{match}(\delta, \widehat{\Delta x}_i) = 0, \quad \bigwedge_{a \in \widehat{\Delta x}_i} \text{adm}(a; \mathcal{Q}_i) = 1. \tag{8}$$

Format-specific adapters normalize volatile metadata and expose semantic objects such as document sections, spreadsheet formulas and ranges, chart bindings, slide elements, database rows, and application records. The match operation evaluates a Contract predicate over these objects, while adm classifies each realized change as required, allowed, or forbidden. Artifact-native checks inspect structure and rendering, including formulas, references, named sections, numerical tolerances, and cross-file consistency. Each result carries a stable evidence identifier consumed by the criterion–evidence map Σ_i. These executable checks produce the integrity, protected-state, and safety gates; the separate module $\mathcal{I}_i$ reports only Harness-controlled initialization, roots, permissions, declared tool versions and availability, state restoration, service, and evidence-capture health h_i^{eval}. Policy-originated malformed calls or tool errors are recorded as process failures when this infrastructure is healthy and affect task status only through an unmet outcome or violated gate.

Reset, evidence, and abstention protocol. Every rollout begins by restoring the same initial workspace state and validating declared roots, permissions, tool versions, and evidence-capture health. The Harness records

$$\mathcal{E}_i = (\tau_i, x_i^0, x_i^{\text{term}}, \widehat{\Delta x}_i, \mathcal{A}_i, \ell_i), \tag{9}$$

where τ_i is the conversation and tool trace, $\mathcal{A}_i$ contains referenced and produced artifact contents or renderings, and ℓ_i contains service audit events. Executable layers have privileged read-only access to both state snapshots. In at most one batched call, the Judge sees only the frozen union $\text{proj}_{\Sigma_i}(\mathcal{E}_i)$ of criterion-tagged evidence blocks, has no live tools, and must cite the permitted block for each decision or abstain when evidence is insufficient. The Harness resolves an abstention to fail when the required state or relation is absent and to invalid when a Harness-controlled evaluation-

health failure prevents a valid decision. Thus the compiled Verifier consumes the complete bundle, $V_i(\mathcal{E}_i) \in \{pass, fail, invalid\}$, rather than an unversioned terminal state alone. The bundle notation above suppresses rollout index r; all terminal-state, trace, criterion, and score quantities are indexed by r when multiple rollouts of task i are compared.

Outcome score and task decision. For an evaluation-valid trajectory ($h_i^{\text{eval}} = 1$), criterion k receives a completion score $p_{i,k} \in [0,1]$ and a binary decision $y_{i,k} = \mathbf{1}[p_{i,k} \geq \gamma_{i,k}]$, where $\gamma_{i,k} \in (0,1]$ is its passing threshold. We normalize the Contract weights as $\bar{w}_{i,k} = w_{i,k} / \sum_{j=1}^{K_i} w_{i,j}$, so that $\bar{w}_{i,k} \geq 0$ and $\sum_k \bar{w}_{i,k} = 1$. The flag $b_{i,k} \in \{0,1\}$ marks critical criteria that must pass. The binary gates $g_i^{\text{integrity}}$, $g_i^{\text{protected}}$, and g_i^{safety} indicate whether integrity, protected-state, and safety checks pass. The outcome score and combined gates are

$$S_{B,i} = \sum_{k=1}^{K_i} \bar{w}_{i,k} p_{i,k}, \qquad H_i = g_i^{\text{integrity}} g_i^{\text{protected}} g_i^{\text{safety}}, \qquad G_i = H_i \prod_{k:b_{i,k}=1} y_{i,k}. \tag{10}$$

Here H_i requires all three binary gates to pass, and G_i additionally requires every critical criterion to pass. For outcome threshold $\tau_B \in (0,1]$, V_i returns $pass$ if and only if

$$G_i = 1 \quad \wedge \quad S_{B,i} \geq \tau_B, \tag{11}$$

and $fail$ otherwise. Partial credit cannot compensate for a failed gate or critical criterion. When a Harness-controlled infrastructure failure prevents evaluation ($h_i^{\text{eval}} = 0$), the result is $invalid$ rather than $fail$.

Process score and threshold selection. The auxiliary trajectory-level process score $S_{A,i,r} \in [0,1]$ (higher is better) is computed offline by frozen deterministic rules over Harness-recorded trace and audit events; it requires no additional Judge call. It penalizes wrong tools or parameters, unrecovered errors, destructive exploration, unsupported state-changing actions, and redundant erroneous calls. It neither rewards brevity nor penalizes exploration required by a long-horizon task, and it never changes the official task status or overrides a hard gate. This trajectory-level score is distinct from VTCA's turn-level trace cost $c_{i,r,t}$, which is computed from predefined trace-event indicators, receives no Judge input, and enters the turn-level advantage estimate explicitly; its definition is given in Appendix B.8. For valid training trajectories, the mixture coefficient of the gated reward in Section 3.3 is restricted to $0 \leq \lambda \leq 0.1$. Criterion thresholds, the overall outcome threshold, and λ are fixed on a held-out development split rather than tuned on test agents.

Mutant families and calibration. Criterion-targeted negative mutants cover six families: local attribute errors in values, units, or dates; structural errors such as missing sections, formulas, or bindings; cross-artifact inconsistencies; protected-state violations; semantic near misses; and adversarial cases such as decoy outputs or artifact-borne prompt injection. Each mutant is a semantic case and version-mapping recipe ρ in a finite, independently validity-checked universe $\mathfrak{R}_i$. For frozen infrastructure-attempt cap B_{inv}, attempt a rematerializes a complete evidence bundle and returns

$$s_i^{(a)}(\rho) = V_i(\text{Mat}_i^{(a)}(\rho)) \in \{pass, fail, invalid\}. \tag{12}$$

Attempts stop at the first $a_i^\dagger(\rho) \leq B_{\text{inv}}$ for which $s_i^{(a_i^\dagger)}(\rho) \neq invalid$. If no such attempt exists, write $a_i^\dagger(\rho) = \infty$. The resolved status and thresholded criterion output are

$$\begin{aligned} \widetilde{s}_i(\rho) &= \begin{cases} s_i^{(a_i^\dagger)}(\rho), & a_i^\dagger < \infty, \\ invalid, & a_i^\dagger = \infty, \end{cases} \\ \widetilde{y}_{i,k}(\rho) &= \begin{cases} y_{i,k}^{(a_i^\dagger)}(\rho), & a_i^\dagger < \infty, \\ \perp, & a_i^\dagger = \infty, \end{cases} \end{aligned} \tag{13}$$

where $\text{Mat}_i^{(a)}(\rho) \in \mathfrak{E}_i$ rematerializes the same semantic case on every attempt and $y_{i,k}^{(a)}$ is criterion k's thresholded output on a non-invalid attempt. All calibration, admission, promotion, and audit metrics, including the Starter–Reference and structural gates, use these resolved quantities; raw invalid attempts are retained only for health reporting. Each negative recipe declares either a target criterion or a Contract-wide failure expectation. For each criterion k, WORKROVER constructs an independently labeled control set $\mathcal{C}_{i,k} \subset \mathfrak{R}_i \times \{0,1\}$ with equal, nonzero frozen quotas for $z = 0$ and $z = 1$. Label $z = 1$ means criterion k should pass on ρ, whereas $z = 0$ means it should fail. Paired criterion agreement is

$$\text{CritAcc}_{i,k}(V_i) = \frac{1}{|\mathcal{C}_{i,k}|} \sum_{(\rho,z)\in\mathcal{C}_{i,k}} \mathbf{1}[\widetilde{y}_{i,k}(\rho) = z]. \tag{14}$$

Task-level negative cases $\mathcal{M}_{i,c}^{-,\mathrm{task}}$ are restricted to mutations for which the frozen Contract entails final failure, through a Contract-wide gate, a critical criterion, or the overall score threshold. Their statistical pools exclude the dedicated leave-one-required-output-out and prescribed protected-state probes used for the structural gates below. Their applicable-family set is $\mathcal{K}_i^{\mathrm{task}} = \{c : |\mathcal{M}_{i,c}^{-,\mathrm{task}}| > 0\}$, which is required to be nonempty and frozen before calibration. For $c \in \mathcal{K}_i^{\mathrm{task}}$, family-level detection is

$$\mathrm{MutScore}_{i,c}(V_i) = \frac{1}{|\mathcal{M}_{i,c}^{-,\mathrm{task}}|} \sum_{\rho \in \mathcal{M}_{i,c}^{-,\mathrm{task}}} \mathbf{1}[\widetilde{s}_i(\rho) = \mathit{fail}]. \tag{15}$$

An infrastructure-level invalid therefore never counts as successful detection. We report the worst-criterion agreement together with the macro-average and worst-family task-level detection, so that many easy numeric mutants cannot hide a weak criterion or relational checker. The nonempty positive pool $\mathcal{M}_i^+$ uses distinct solution paths or semantics-preserving transformations and is summarized by

$$\mathrm{AltPass}_i(V_i) = \frac{1}{|\mathcal{M}_i^+|} \sum_{\rho \in \mathcal{M}_i^+} \mathbf{1}[\widetilde{s}_i(\rho) = \mathit{pass}]. \tag{16}$$

Before candidate repair, all final-status and criterion-control recipe uses are jointly partitioned at the recipe-seed level into a candidate-visible calibration shard, a one-time initial-admission shard, a sealed audit shard, and at most $M_{\max}$ preallocated one-time promotion shards. Thus a recipe may carry both label types only in the same shard and never crosses visibility classes; the initial-admission shard is permanently retired after use. Criterion controls have equal, nonzero $z = 0$ and $z = 1$ quotas for every criterion within every shard. Every shard also has a nonzero quota for each $c \in \mathcal{K}_i^{\mathrm{task}}$ and a nonzero positive-case quota. A package whose finite recipe bank cannot support these disjoint quotas is rejected before repair, or receives a smaller development-frozen $M_{\max}$. Mutant-family definitions, generators, labeling rules, per-family quotas, the maximum number of repair rounds, and thresholds $\alpha_{\mathrm{crit}}, \alpha_-, \alpha_+ \in (0, 1]$ are selected on development tasks and then frozen. Let A_i^{req} indicate that every leave-one-required-output-out case yields a healthy fail, let A_i^{protect} indicate that all prescribed protected-state mutants yield healthy fail. These deterministic structural gates sit outside the statistical shards, use no sealed audit case, and are rematerialized after every repair. For shard $s \in \{\mathrm{cal}, \mathrm{adm}, \mathrm{audit}\}$, a superscript s restricts $\mathrm{CritAcc}$, $\mathrm{MutScore}$, $\mathrm{AltPass}$, or $\mathrm{UnresolvedInvalid}$ to that shard; the fixed argument V_i is omitted. An infrastructure-invalid attempt is retried for the unchanged case and candidate only up to a frozen cap; $\mathrm{UnresolvedInvalid}_i^s$ is the fraction of cases with $\widetilde{s}_i(\rho) = \mathit{invalid}$. Such a case contributes neither a correct criterion label nor a detected negative or accepted positive. The raw invalid-attempt rate is reported separately and does not itself determine admission. The admission predicate is

$$\begin{aligned} \mathrm{Retain}_i = \mathbf{1}[&\widetilde{s}_i(\rho_i^0) = \mathit{fail} \ \wedge \ \widetilde{s}_i(\rho_i^\star) = \mathit{pass} \\ &\wedge \ A_i^{\mathrm{req}} = 1 \ \wedge \ A_i^{\mathrm{protect}} = 1 \\ &\wedge \ \min_k \mathrm{CritAcc}_{i,k}^{\mathrm{adm}} \geq \alpha_{\mathrm{crit}} \\ &\wedge \ \min_{c \in \mathcal{K}_i^{\mathrm{task}}} \mathrm{MutScore}_{i,c}^{\mathrm{adm}} \geq \alpha_- \\ &\wedge \ \mathrm{AltPass}_i^{\mathrm{adm}} \geq \alpha_+ \ \wedge \ \mathrm{UnresolvedInvalid}_i^{\mathrm{adm}} = 0]. \end{aligned} \tag{17}$$

Here ρ_i^0 and $\rho_i^\star$ rematerialize the Starter and Reference cases. Only failures in the candidate-visible calibration set may trigger pre-admission repair. Such cases may directly guide only the evidence projection, checker, or compiler factor. A Context, Task, or Harness factor is activated only when proposal-disjoint, instruction-visible evidence independently establishes the same typed mismatch; its proposal receives no mutant content, label, or hidden Contract field. The resulting candidate-stage FCR invocation freezes all direct proposals, enumerates their compatible closures, and returns a selected cell to full calibration before its one-time initial admission. A Contract-field change creates a new Contract version, discards and independently relabels all shards, and restarts the full procedure; an intended-outcome or state change additionally requires a fresh Demonstrator rollout and is not an FCR repair. The concealed admission set returns one binary gate, is retired after use, and never exposes case-level feedback; failure after the capped infrastructure retry rule rejects the candidate and cannot trigger FCR. The mutant generator changes only after independent review finds a calibration case invalid or mislabeled, which also restarts the procedure with fresh seeds. The sealed audit set never affects repair, threshold selection, or admission and is opened only for final reporting. Cross-file necessity is tested separately by ablating each declared input binding and checking whether its associated outcome or evidence lineage can still be justified.

Held-out mutants and blinded human adjudication estimate false-pass, false-reject, and Judge-instability rates. Repeated trials report Pass@k, meaning at least one success in k runs, and Pass^k, meaning all k runs succeed, thereby separating occasional from consistent completion.

Serialization and post-hoc QA. Each Admitted Package P_i contains the instruction, Starter WorkEnv, Reference State, rubric, evidence, and accepted traces. Serialized packages use content-addressed maps over changed objects so that Starter, Reference, and mutant states can be reconstructed without duplicating unchanged material. Post-hoc structural and lineage QA validates record shape, message roles, loss masks, turn boundaries, encoding, manifest completeness, and parent–child lineage. These checks determine training eligibility and reproducibility, but never substitute for outcome verification.

B.4 Full FCR Mechanics and Promotion Rules

This subsection expands the repair boundaries, anchor construction, factorial estimators, and promotion rules summarized in Section 3.2. Figure 13 illustrates maintenance-stage quarter-close repair with C fixed. Proposals bind T to the public batch manifest, restore H's declared shared checkpoint, and enforce existing row-lineage and foreign-exchange requirements in V.

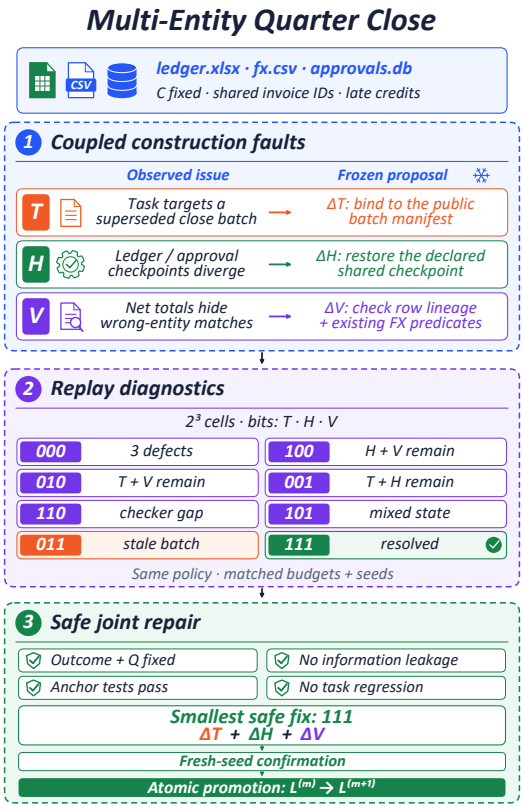

Figure 13: **Multi-entity quarter close.** Joint $T+H+V$ repair under a fixed Contract.

Repair boundaries and stage-specific freeze. The typed trigger ledger records Context dependencies and versions, Task requirements and preconditions, Harness initialization and evidence capture, and Verifier criteria and evidence surfaces. It identifies suspected components U_i and the subset F_i for which one admissible direct revision is available. Exactly one proposal per factor in F_i is frozen before any factorial replay is observed. Components without direct proposals receive no direct edits but may undergo version updates induced by deterministic dependency closure. A Context revision may repair a missing or inconsistent dependency; a Task revision may clarify an instruction, binding, or precondition; a Harness revision may repair initialization, permissions, tool bindings, recovery, or raw-evidence capture; and a Verifier revision may repair canonicalization, a typed matcher, checker implementation, or evidence projection.

Candidate-stage FCR freezes the unadmitted parent candidate and preserves $\mathcal{Q}_i^{(0)}$ before grounding or $\mathcal{Q}_i$ afterward; a selected cell returns to the earliest affected construction gate. Maintenance-stage FCR instead freezes the admitted generation-m parent and may promote a closed revision to generation $m+1$. In either mode, ΔT_i preserves task intent, required outcome, and the intended state transition; once V_i exists, ΔV_i holds Contract fields, weights, critical flags, hard gates, and τ_B fixed; and ΔH may restore only behavior already declared by the shared Contract. An added agent-visible tool, schema, memory, or planning aid requires a separate workspace-level Harness release. Changing an applicable Contract field or intended outcome forks a new candidate or package rather than repairing the frozen parent. If deterministic closure requires a semantic edit outside the frozen candidates, that edit likewise starts a new candidate or construction generation.

Agent-visible Context, Task, and Harness candidates may not introduce tokens, artifacts, relations, or evidence derived from an available Reference State $x_i^\star$, mutants, or hidden fields of the applicable Contract anchor beyond what the parent exposed. A blinded reviewer checks intent and outcome equivalence. A frozen solver panel that did not generate the candidates requires changes in solver success or effort to stay within development-frozen simplification margins.

Contract anchor banks. Before Verifier compilation, candidate-stage FCR uses the frozen pre-grounding Contract, typed construction predicate, and proposal-disjoint Demonstrator replays; no Verifier score or anchor bank participates in cell selection. The selected cell must subsequently traverse Phase III and initial admission. Once a Verifier exists, FCR prevents it from validating itself by constructing its anchors before candidate generation. Let $\mathcal{M}_i^{-,\mathrm{task}} = \bigcup_{c\in\mathcal{K}_i^{\mathrm{task}}} \mathcal{M}_{i,c}^{-,\mathrm{task}}$. The final-status and criterion-level banks are

$$\mathcal{B}_i^{\mathrm{bank}} = \{(\rho, \mathit{fail}) : \rho \in \mathcal{M}_i^{-,\mathrm{task}}\} \cup \{(\rho, \mathit{pass}) : \rho \in \mathcal{M}_i^{+}\}, \tag{18}$$

$$\mathcal{C}_i^{\mathrm{bank}} = \{(\rho, k, z) : (\rho, z) \in \mathcal{C}_{i,k}, 1 \le k \le K_i\}. \tag{19}$$

Both banks are jointly partitioned at the recipe–seed level into a candidate-visible calibration shard, a permanently retired initial-admission shard, a sealed audit shard, and preallocated one-time promotion shards

$$\{\mathcal{B}_{i,\ell}^{\mathrm{prom}}, \mathcal{C}_{i,\ell}^{\mathrm{prom}}\}_{\ell=1}^{M_{\max}}. \tag{20}$$

The maximum number of repair invocations, split, quotas, and thresholds are frozen before any proposal is made. A recipe carrying both label types remains within one visibility class. The candidate may use only the calibration shard and receives one binary result, without case-level feedback, from invocation ℓ's promotion pair, which is retired after use. The sealed audit shard affects neither repair nor selection and is opened only for reporting.

Each ρ is a semantic case plus a version-mapping recipe. For cell $\boldsymbol{\chi}$, it rematerializes an evidence-complete bundle through $\overline{H}(\boldsymbol{\chi}_{-V})$ without changing the semantic label. Under the frozen capped-retry rule, $\widetilde{s}_i(\boldsymbol{\chi};\rho)$ is the first healthy status returned by $\overline{V}_i(\boldsymbol{\chi})$, or *invalid* if no healthy attempt is obtained. Starter–Reference, required-output, and protected-state structural checks are independently rematerialized and do not belong to the statistical banks.

Harness screening before replay. Before materialization or matched policy replay, the Harness applies frozen deterministic checks to each proposed cell's planned dependency closure. These checks reject only established conflicts in dependencies, versions, bindings, or frozen requirements. Closure-induced updates are included, so an unselected proposal does not imply an unchanged component. Uncertain cases proceed to materialization and full compatibility checks. Pruning uses only these structural checks; policy success, candidate-Verifier scores, and repair benefit do not affect this stage. All rejected cells and reasons remain in the compatibility ledger; their responses are unavailable and are not imputed. Remaining cells must still pass every applicable admission or promotion gate. Factorial contrasts use the complete compatible subcubes specified below.

Matched factorial estimators. For R matched seeds, let $h_{i,r}^{\mathrm{eval}}(\boldsymbol{\chi})$ denote evaluation health, $\mathcal{R}_i^{\mathrm{val}}(\boldsymbol{\chi}) = \{r : h_{i,r}^{\mathrm{eval}}(\boldsymbol{\chi}) = 1\}$, and $P_{i,r}(\boldsymbol{\chi}) = \mathbf{1}[\mathrm{status}_{i,r}(\boldsymbol{\chi}) = \mathit{pass}]$. For fixed $\boldsymbol{\chi}_{-V}$ and seed r, Verifier alternatives score the same recorded execution evidence offline and inherit the same Harness-controlled health flag: $h_{i,r}^{\mathrm{eval}}(0_V, \boldsymbol{\chi}_{-V}) = h_{i,r}^{\mathrm{eval}}(1_V, \boldsymbol{\chi}_{-V})$. Health is averaged over all scheduled seeds, whereas gate, score, and pass are averaged only over healthy executions:

$$\mu_{i,\mathrm{health}}(\boldsymbol{\chi}) = R^{-1} \sum_{r=1}^{R} h_{i,r}^{\mathrm{eval}}(\boldsymbol{\chi}), \tag{21}$$

$$\mu_{i,q}(\boldsymbol{\chi}) = \frac{1}{|\mathcal{R}_i^{\mathrm{val}}(\boldsymbol{\chi})|} \sum_{r\in\mathcal{R}_i^{\mathrm{val}}(\boldsymbol{\chi})} q_{i,r}(\boldsymbol{\chi}), \quad q \in \{\mathit{gate}, \mathit{score}, \mathit{pass}\}, \tag{22}$$

where $q_{i,r}$ is respectively $G_{i,r}$, $S_{B,i,r}$, or $P_{i,r}$; the second mean is undefined when no replay is healthy. Anchor agreement is $\mathrm{Acc}_i(\boldsymbol{\chi};\mathcal{B}) = |\mathcal{B}|^{-1}\sum_{(\rho,y)\in\mathcal{B}}\mathbf{1}[\widetilde{s}_i(\boldsymbol{\chi};\rho) = y]$. The common response is

$$\phi_{i,q}(\boldsymbol{\chi}) = \begin{cases} \mu_{i,\mathrm{health}}(\boldsymbol{\chi}), & q = \mathit{health}, \\ \mu_{i,q}(\boldsymbol{\chi}), & q \in \{\mathit{gate}, \mathit{score}, \mathit{pass}\}, \\ \mathrm{Acc}_i(\boldsymbol{\chi};\mathcal{B}_i^{\mathrm{audit}}), & q = \mathit{anchor}. \end{cases} \tag{23}$$

After freezing the compatibility ledger but before revealing response values, FCR enumerates all inclusion-maximal complete compatible subcubes $\mathcal{S}_i^{\mathrm{cube}}$. For a subcube whose varying factors are $U \subseteq F_i$, fixed factors are $\boldsymbol{\xi}$, $u = |U|$, and cell set is $\mathcal{X}(U,\boldsymbol{\xi})$. Gate, score, and pass responses are recomputed on the shared complete-case seeds $\mathcal{R}_i^{\mathrm{cc}}(U,\boldsymbol{\xi}) = \bigcap_{\boldsymbol{\chi}\in\mathcal{X}(U,\boldsymbol{\xi})}\mathcal{R}_i^{\mathrm{val}}(\boldsymbol{\chi})$; health retains all R scheduled seeds. For $a, b \in U$, FCR reports

$$\mathrm{ME}_{a,q}^{U,\boldsymbol{\xi}} = 2^{-(u-1)}\sum_{\boldsymbol{\chi}_{U\setminus\{a\}}}\left[\phi_{i,q}(1_a,\boldsymbol{\chi}_{U\setminus\{a\}};\boldsymbol{\xi}) - \phi_{i,q}(0_a,\boldsymbol{\chi}_{U\setminus\{a\}};\boldsymbol{\xi})\right], \tag{24}$$

$$\begin{aligned}\mathrm{INT}_{a,b,q}^{U,\boldsymbol{\xi}} = 2^{-(u-2)}\sum_{\boldsymbol{\chi}_{U\setminus\{a,b\}}}\big[&\phi_{i,q}(1_a,1_b,\boldsymbol{\chi}_{U\setminus\{a,b\}};\boldsymbol{\xi}) - \phi_{i,q}(1_a,0_b,\boldsymbol{\chi}_{U\setminus\{a,b\}};\boldsymbol{\xi}) \\ &- \phi_{i,q}(0_a,1_b,\boldsymbol{\chi}_{U\setminus\{a,b\}};\boldsymbol{\xi}) + \phi_{i,q}(0_a,0_b,\boldsymbol{\chi}_{U\setminus\{a,b\}};\boldsymbol{\xi})\big].\end{aligned} \tag{25}$$

A contrast is unavailable if any required response is missing; gate, score, and pass contrasts additionally require a nonempty common seed set. All pre-enumerated subcubes are reported with the preregistered multiplicity correction; none is chosen by its observed effect. On every complete subcube with $V \in U$, shared health flags imply $\mathrm{ME}_{V,\mathrm{health}}^{U,\boldsymbol{\xi}} = 0$ and $\mathrm{INT}_{a,V,\mathrm{health}}^{U,\boldsymbol{\xi}} = 0$ for $a \in U \setminus \{V\}$. These are structural identities of the replay protocol. For scoring contrasts containing V, sealed final-status and criterion agreement are the reporting metrics because official pass and $S_{B,i}$ change measurement implementation with χ_V. These are candidate-specific compatibility contrasts, not universal component effects.

Complete admissibility and closure rules. The trigger-specific predicate $\mathrm{Resolve}_i$ is frozen before candidate generation and contains typed ledger predicates plus independent runtime-health checks; it excludes candidate-Verifier scores, $S_{B,i}$, and policy success. In every invocation, a closed cell must satisfy (i) $\mathrm{Cmp}_i(\boldsymbol{\chi}) = 1$ and resolve the trigger on proposal-disjoint replay seeds and (ii) pass the information-boundary, blinded-equivalence, and solver-panel gates. For a Phase II candidate trigger, these conditions plus a fresh valid Demonstrator Reference define $\mathcal{Z}_i^{\mathrm{safe}}$; the selected cell then proceeds through the ordinary Phase III gates. For a candidate-stage Phase III or maintenance-stage trigger, a cell additionally requires (iii) a healthy-fail Starter, a healthy-pass fresh Demonstrator Reference, and $A_i^{\mathrm{req}} = A_i^{\mathrm{protect}} = 1$; (iv) on the one-time promotion pair,

$$\begin{aligned}&\mathrm{UnresolvedInvalid}_{i,\ell}^{\mathrm{prom}}(\boldsymbol{\chi}) = 0, \qquad \min_k \mathrm{CritAcc}_{i,k,\ell}^{\mathrm{prom}}(\boldsymbol{\chi}) \geq \alpha_{\mathrm{crit}}, \\ &\min_{c\in\mathcal{K}_i^{\mathrm{task}}} \mathrm{MutScore}_{i,c,\ell}^{\mathrm{prom}}(\boldsymbol{\chi}) \geq \alpha_-, \qquad \mathrm{AltPass}_{i,\ell}^{\mathrm{prom}}(\boldsymbol{\chi}) \geq \alpha_+;\end{aligned} \tag{26}$$

and (v) it causes no regression in affected retained packages. These additional conditions define $\mathcal{Z}_i^{\mathrm{safe}}$ once a Verifier exists. All quantities in (iii)–(iv) are recomputed on the exact closed tuple.

Closure propagates a Context change through dependent Task and Harness bindings, reference traces, and Verifier surfaces; a Task change requires a fresh Demonstrator Reference State and Verifier compilation; and a Verifier change is recalibrated on the visible calibration banks before its concealed promotion test. FCR invocations are serialized within the current workspace. When a direct or induced Harness diff affects a shared capability or evidence surface, $\Gamma_i(\boldsymbol{\chi})$ expands to all retained packages in the workspace that depend on it, and admissibility is aggregated over their replays. The lexicographic rule in Section 3.2 is applied only after these gates, followed by fresh-seed confirmation. A candidate-stage cell returns atomically to the earliest affected construction step and remains unaccepted until it clears initial admission; only a maintenance-stage cell is promoted as generation $m + 1$ of an admitted package. If selection or fresh-seed confirmation fails, candidate-stage FCR discards the unadmitted candidate; maintenance-stage FCR retains the parent and quarantines the unresolved trigger. Initial-admission failure rejects the candidate without repair feedback.

B.5 Evaluation Protocol for Recursive Environment Evolution

This subsection specifies the sampling, baselines, safety ablations, and statistical analysis supporting RQ5–RQ6. The protocol is frozen before any repair outcome is observed. Candidate-stage synthe-

sis and post-admission maintenance use disjoint cohorts and denominators. Within maintenance, SRR is reported separately for natural and controlled cohorts: the former estimates performance on rollout-discovered failures, whereas the latter provides known component-level ground truth. Appendix C.6 centralizes the maintenance metric definitions, denominators, missing-data rules, and relative-cost accounting.

Candidate-stage synthesis cohort. Before any repair, a Work-Context-stratified random stream of N_0 initial Task Candidates is frozen together with a total post-attempt repair budget per candidate. Candidate identifiers, including candidates later regenerated in full, remain fixed in the denominator of

$$\mathrm{SynthPass}_B(A) = \frac{1}{N_0} \sum_{i=1}^{N_0} \mathbf{1}[\mathrm{Admit}_i(A; B) = 1], \tag{27}$$

where $\mathrm{Admit}_i(A; B)$ records whether method A brings initial candidate i through its first admission within repair budget B. At $B = 0$, this reduces to the shared first-attempt pass rate; post-admission maintenance outcomes never enter this denominator. We first run Phase II and III without repair to measure the common first-attempt pass rate and record the earliest failed gate. Proposal-disjoint diagnostics classify each failure as policy-only, infrastructure-level, Contract-preserving and repairable, or non-repairable. Repair methods receive identical candidate-visible evidence and matched proposal, replay, Verifier-call, and wall-clock caps. A repaired candidate traverses every downstream gate and may use the concealed initial admission set exactly once; that binary result supplies no repair feedback. We report final synthesis pass rate over all N_0 candidates and conditional recovery over the frozen, initially failed repairable subset. Neither quantity is pooled with post-admission SRR or retained validity.

Maintenance-stage sampling frame and experimental units. The full 10K-task corpus is used for the static coverage census and as the sampling frame, but full factorial replay is not run on every package. Within the held-out construction partition, task packages are placed in a random screening order stratified by Work Context, workspace, artifact composition, difficulty, and Harness capability. An independent trigger rollout uses seeds disjoint from initial admission, proposal generation, replay, promotion, and final audit. Screening continues through the frozen order until 300 confirmed natural construction mismatches are obtained; if the partition contains fewer, all confirmed mismatches are used and the shortfall and screening yield are reported. No workspace crosses a construction, training, or evaluation split.

Two reviewers, blind to repair proposals and outcomes, independently classify each trigger as a Context, Task, Harness, Verifier, policy, or infrastructure failure and may assign multiple construction components. A third reviewer adjudicates disagreements. A parent that is Contract-consistent, reproducibly solvable by the Demonstrator, and healthy under the Harness is a policy-failure control rather than an FCR trigger. We additionally retain 60 healthy or policy-failure controls to measure false repair. Replay cells and random seeds are repeated measurements nested within a trigger; they never increase the nominal sample size.

Controlled defect cohort. We draw admission-clean, reproducibly healthy parents from workspaces disjoint from the natural cohort and inject 260 Contract-preserving failures: 20 examples for each of the four single-component faults (80), 20 for each of the six component pairs (120), 10 for each of the four triples (40), and 20 four-component faults. Injection labels and recipes are unavailable to the ledger, proposal models, solver panel, and repair methods. Every injected child must fail its intended typed predicate while its unmodified parent retains its recorded admission pass and continues to pass fresh health and Reference checks. Table 8 records the admissible fault families and invariants.

Candidate-controlled and end-to-end tracks. For the candidate-controlled comparison, a single proposal stage emits at most one direct revision for each active factor, and the proposals are frozen before any replay result is visible. Local-only, Greedy-sequential, Apply-all, FCR, and the Oracle-cell analysis receive this identical proposal set. This track separates proposal coverage—whether any safely resolving cell exists—from selection quality. In the end-to-end comparison, each deployable method generates its own repairs under the same proposal-token, tool-call, and total rollout budgets. Full-regeneration must preserve the frozen Contract and is subject to the same intent, information-boundary, solver-effort, anchor, and cross-task regression checks. End-to-end methods receive the same diagnostic evidence and component ledger; only their proposal generation and selection rules differ. Injection labels, known inverse repairs, and final audit outcomes are withheld from all deployable methods. Table 10 gives the exact decision rules.

Table 8: Controlled FCR fault library. Higher-order cases compose these faults without changing the Evaluation Contract or intended state transition.

Factor	Injected construction mismatch	Frozen repair boundary
Context	Missing declared dependency, stale version mapping, inconsistent relation, or conflicting artifact binding.	No answer-derived artifact or hidden Contract field is introduced; professional intent and required outcome remain unchanged.
Task	Ambiguous or stale input/output binding, missing stated precondition, or instruction–manifest inconsistency.	The required outcome and intended state transition remain fixed; a simplification or new outcome is a fork, not a repair.
Harness	Reset, mount, permission, tool binding, recovery, or raw-evidence capture mismatch.	The candidate may restore only declared behavior; it may not add an agent-visible tool, memory, schema, or planning aid.
Verifier	Canonicalization, typed matcher, checker implementation, or evidence projection mismatch.	$\mathcal{Q}_i$, its criterion weights, hard gates, critical flags, labels, and τ_B remain fixed.

Diagnostic accuracy and proposal coverage. The diagnostic is evaluated before repair proposals are generated. Its binary decision is construction mismatch versus healthy/policy/infrastructure cases; its component output is U_i. Against independently adjudicated labels, report the detection confusion matrix, per-component precision/recall, exact-set accuracy, abstentions, and false repair on healthy or policy-failure controls. Infrastructure failures remain a separate class. The 300/260/60 cohort sizes above are sampling targets; their realized counts and diagnostic results are recorded separately in Table 9. Because the cohorts are enriched for defects, pooled precision is not a natural-prevalence estimate. Population estimates require the full screening denominator and sampling weights; controlled-cohort results remain separate.

For a frozen proposal set, let $\Omega_i = 1$ if at least one candidate cell safely resolves the trigger under independent reporting checks, and let $S_i = 1$ if the selected cell passes fresh confirmation and final audit. Report proposal coverage $\sum_i \Omega_i/N$ and conditional selection success $\sum_i \Omega_i S_i / \sum_i \Omega_i$ alongside unconditional SRR. Cells not fully checked make coverage unknown: report the confirmed lower bound and the upper bound including unknown triggers rather than coding them as zero. The analysis-only oracle identifies the best cell already present; it cannot repair a missing proposal. Component recall measures the upstream ledger, while proposal coverage measures the existence of a usable repair.

On controlled defects, record whether the known fault support was omitted from U_i or F_i, whether included proposals failed to restore a valid configuration, and whether selection or fresh confirmation failed despite a resolving cell. Known inverse repairs are analysis-only controls and must themselves pass the frozen gates. On natural defects, unresolved causes may remain unknown; a failed search is not proof that no valid repair exists. These diagnostics are needed to attribute the gains to replay and selection rather than proposal generation alone.

Table 9: FCR diagnostic and proposal-coverage reporting. Sampling targets are not realized counts.

Cohort	Target/actual N	Detection recall	Component P/R	Coverage	Selection $\mid \Omega = 1$	False repair
Natural defects	300/300	95.00	93.00/91.00	80.00	91.25	–
Controlled defects	260/260	97.69	94.96/86.92	87.69	92.54	–
Healthy/policy controls	60/60	–	–	–	–	1.67

Replay and confirmation budget. The main protocol uses $R = 5$ matched replay seeds per distinct Context–Task–Harness execution configuration; compatible Verifier variants score the same evidence offline. A preregistered 20% stratified subset is repeated with $R = 10$ to test seed sensitivity. Fresh confirmation uses new seeds, and the three-generation longitudinal experiment uses distinct preallocated promotion shards. All attempted runs, including capped infrastructure retries, count toward the reported budget. We report the number of scored cells and the smaller number of unique executions separately. Relative cost counts total model input and output tokens consumed by proposals, model calls during rematerialization, replay, and fresh confirmation, including Verifier/Judge calls for calibration, offline scoring, and promotion, plus solver-panel and affected-workspace validation calls used in repair decisions. Failed attempts and retries are charged. Deterministic environment CPU use and tool wall-clock time are recorded separately and are not converted to tokens. Independent final reporting audits are accounted for separately from repair tokens; Appendix C.6 specifies the ledger and normalization required for the relative-cost column.

Table 10: Repair baselines for the recursive environment evolution experiment. All deployable methods use matched parent lineages and equal total proposal and replay budgets.

Method	Decision rule	Purpose
Parent / no repair	Replay the unchanged parent under the common seed schedule.	Establish unresolved-trigger and natural-recovery rates.
Local-only	Apply only the highest-confidence component proposal fixed by the ledger before replay.	Test whether isolated repair is sufficient.
Greedy-sequential	Visit components in a preregistered ledger order, retaining an edit when its next matched replay improves the frozen resolution predicates.	Test sequential credit and path dependence.
Apply-all	Apply every active frozen proposal and materialize its exact closure.	Test whether factorial selection is needed beyond proposal union.
Full-regeneration	Regenerate the complete package under the frozen Contract and matched total budget.	Test whether localized joint repair better preserves intent and cost.
FCR	Enumerate compatible closed cells, apply concealed gates, and choose the lexicographically minimal admissible cell.	Complete method.
Oracle cell	Retrospectively choose the safest admissible cell after opening all replay and audit outcomes.	Analysis-only upper bound; never deployed.

Factorial scope and absolute resource accounting. Each invocation has at most one direct proposal for each of $\{C, T, H, V\}$, hence $0 \leq d_i \leq 4$ and at most 16 cells; before Verifier compilation $d_i \leq 3$. With $d_i = 0$, there is no repair candidate beyond the parent. This bounds direct-factor enumeration, not dependency-closure size, affected-task count, or model execution cost. No claim is made that exhaustive replay scales to arbitrarily many independent proposals.

For each trigger, record the nested counts of proposed, materializable, replay-evaluable, admissible, and independently confirmed cells. Also report combinations rejected by Harness screening and their reasons; include screening CPU and elapsed time in the closure stage. Assign each attempted cell one final status and failure reason, so terminal status counts are mutually exclusive. Report d_i, closure size, affected tasks, safe-cell count, distinct execution configurations, retries, and budget exhaustion. Verifier variants reuse execution evidence only when their closed Context–Task–Harness states and seed are identical; score-cell counts must not be substituted for execution counts.

Absolute costs are split into diagnosis, proposal generation, closure and materialization, replay, scoring, cross-task checks, and fresh confirmation. For every stage, report model calls, input/output tokens, deterministic CPU seconds, accelerator device-hours with hardware and concurrency, storage, and elapsed wall time. Stage timings may overlap and are not summed to manufacture end-to-end latency. Report medians and P95 by d_i, plus total resources over all attempted triggers, charging unsuccessful and retried attempts. Independent reporting audits have their own cost line.

Table 11: FCR factor-count and absolute-cost breakdown. Wall time is per trigger.

d	Cell cap	N	Materializable	Safe cells	Executions	Model calls	Wall median/P95	Budget exhausted
0	1	84	1.0	0.0	5.0	470	8/20	0.0
1	2	156	1.9	0.9	12.0	1140	18/45	1.3
2	4	240	3.5	1.5	18.0	1740	30/80	3.3
3	8	100	6.4	2.0	28.0	2750	52/150	8.0
4	16	40	11.5	2.6	44.0	4440	90/280	15.0

Necessity measurements. For controlled cases, component attribution is evaluated against the injected direct-factor support using exact-set accuracy, per-factor precision and recall, macro-F1, and unnecessary-component edits. Closure-induced version changes are reported separately and are not counted as direct-attribution errors. For a safely resolving selected cell $\boldsymbol{\chi}_i^\star$, joint necessity holds when $|\operatorname{supp}(\boldsymbol{\chi}_i^\star)| > 1$ and no strict subset of that direct support passes the same frozen resolution, information-boundary, anchor, regression, and confirmation conditions. We report this joint-necessity rate overall and by factor pair, active-factor count, artifact class, and difficulty. Main and interaction effects use only the complete compatible subcubes defined in Appendix B.4; the

denominator, common-seed count, and unavailable contrasts are always reported. Tables 12 and 13 separate attribution quality from measured component compatibility.

Table 12: Controlled-defect attribution and repair results by injected direct support size. Allocated N follows the frozen sampling design. Outcome entries are point estimates; confidence intervals are not shown in this table. Extra edits exclude deterministic closure-induced version changes.

Injected support	Allocated N	Ledger macro-F1↑	Exact support↑	Extra edits↓	Joint necessary↑	FCR SRR↑
Single component	80	0.950	0.900	0.025	–	90.0
Component pair	120	0.923	0.750	0.075	0.637	75.8
Component triple	40	0.877	0.600	0.125	0.885	65.0
All four	20	0.857	0.350	0.000	1.000	65.0
Overall	260	0.908	0.742	0.062	0.465	77.7

Table 13: Pairwise compatibility summary over preregistered complete subcubes. Interactions are matched-seed contrasts from Appendix B.4; unavailable subcubes remain in the reported denominator. Health entries marked † are structural zeros implied by shared execution evidence across Verifier alternatives. Adjusted p uses Holm–Bonferroni correction over the six pairs.

Pair	Complete cubes	$\mathrm{INT}_{\mathrm{health}}$	$\mathrm{INT}_{\mathrm{anchor}}$	Joint necessary	Unavailable↓	Adjusted p
$C \times T$	54	0.18	0.08	0.64	10.0	0.290
$C \times H$	56	0.24	0.06	0.71	6.7	0.120
$C \times V$	51	$0^{\dagger}$	0.14	0.48	15.0	0.830
$T \times H$	55	0.20	0.10	0.67	8.3	0.250
$T \times V$	52	$0^{\dagger}$	0.18	0.55	13.3	0.830
$H \times V$	56	$0^{\dagger}$	0.17	0.74	6.7	0.087

Safety audits and analysis-only ablations. The primary safety outcomes are sealed final-status agreement, minimum and macro criterion agreement, negative-mutant detection, positive-alternative acceptance, information-boundary violations, blinded task-intent equivalence, solver-success and effort drift, protected-state violations, false repair on controls, and regressions in affected retained packages. At least 200 parent–repair pairs are sampled across methods, factors, and outcomes for two independent blinded reviewers, with adjudication of disagreements. Intent drift is estimated over the reviewed pairs with an adjudicated label, rather than assuming that every maintenance trigger has a human label. Reporting gives the reviewed, adjudicated, and unknown-label counts by method and cohort; unknown labels are not treated as evidence of intent preservation. This rate describes the adjudicated audit sample and is not automatically a population prevalence estimate. The ablations in Table 14 are executed offline and their outcomes are reported in Table 15; no package they select is released or used for training.

Table 14: Analysis-only FCR safeguard ablations and their targeted failure modes.

Ablation	Targeted diagnostic
No exact closure	Stale bindings, references, compiled Verifiers, or induced Harness versions that direct-file replacement fails to rematerialize.
Unmatched replay seeds	Selection instability and inflated variance caused by comparing cells under different stochastic conditions.
Calibration-only / reused promotion cases	Apparent Verifier improvement that does not transfer to the permanently sealed audit banks.
No information-boundary gate	Answer-derived tokens, artifacts, relations, or hidden evaluation fields that increase pass rate by leakage.
Score-max selection	Larger semantic diffs and calibrated solver-effort drift caused by choosing the highest observed rollout score instead of the minimal safe repair.
No cross-task regression	Escaped failures in retained packages sharing an affected Harness capability or evidence surface.
No fresh confirmation	Promotions that succeed on replay seeds but fail on a new seed allocation.

Longitudinal and held-out-policy audits. For three construction generations, 50–60 workspaces are followed from the same frozen starting release. Each generation receives an independent trigger seed allocation and its one-time promotion shard; the selected closed tuple is the next generation's parent. We report confirmed construction defects per 100 trigger rollouts, SRR, quarantine and rollback rates, sealed-audit acceptance, retained validity (the share of the original admitted cohort

Table 15: Outcomes of the analysis-only safeguard ablations in Table 14. Raw resolution is shown beside SRR so that apparent gains obtained through leakage, intent drift, Verifier weakening, or cross-package Harness regressions remain visible. No ablation-selected package is released or used for training.

Variant	Raw resolve↑	SRR↑	Info. violation↓	Intent drift↓	Audit false pass↓	Task reg.↓
Full FCR	73.0	70.3	0.7	1.3	1.0	1.1
No exact closure	65.0	53.0	0.8	2.0	6.0	6.5
Unmatched seeds	76.0	64.0	0.8	1.8	3.0	2.5
Reused promotion cases	81.0	60.0	1.0	2.0	9.0	2.0
No information boundary	82.0	59.0	14.0	5.0	6.0	2.0
Score-max selection	77.0	68.0	0.8	3.0	2.0	1.8
No cross-task regression gate	78.0	65.0	0.8	1.5	1.5	8.0
No fresh confirmation	78.0	63.0	0.8	1.5	2.5	2.0

that remains audit-clean), closure size, and marginal cost by generation. A stratified set of at least 150 promoted and retained parents is then executed by policies that did not produce the trigger or proposals. The audit reports evaluation health, pass rate, criterion completion, solver effort, and agent ranking stability, testing whether repair transfers beyond the rollout policy without claiming that FCR itself trains that policy. Table 16 preserves the exact generation-level estimates underlying Figure 7(c).

Table 16: Three-generation FCR audit. Each row uses its independently preallocated trigger seeds and promotion shard. Closure size counts all direct and induced component versions; cost is marginal to retaining the parent.

Generation	Workspaces	Defects / 100	SRR↑	Audit OK↑	Retained valid↑	Quarantine↓	Closure size↓	Marginal cost↓
$m = 1$	55	18.0	70.3	93.8	98.9	29.7	3.1	1.00
$m = 2$	55	13.0	72.4	94.6	98.2	27.6	2.8	0.85
$m = 3$	55	10.0	71.8	94.2	97.6	28.2	2.7	0.78

Statistical analysis and reporting split. Candidate-stage $\mathrm{SynthPass}_B$ and maintenance-stage SRR are reported as distinct stage-specific endpoints. Within maintenance, the natural cohort is the primary population estimate; controlled defects and healthy controls are reported separately rather than pooled into its SRR estimate. Audit OK and Task reg. use the frozen union of natural and controlled triggers, excluding controls, and weight triggers equally; this is weighting cohort rates by their sample sizes, not averaging the two rates equally. Relative token cost uses the same union within each comparison track. These combined diagnostics describe the fixed evaluation mixture rather than the natural-failure population. Intent drift instead uses the adjudicated audit sample from the two cohorts, with its method- and cohort-specific denominators and unknown counts reported as specified above. Binary method comparisons are paired on trigger and seed schedule; paired intent-drift comparisons require pairs with adjudicated labels for both methods. Where shown, confidence intervals use the 95% workspace-clustered bootstrap procedure, with the Controlled Parent boundary exception defined in Appendix C.6. Paired randomization tests use Holm–Bonferroni correction over preregistered baseline and factor-interaction families. Continuous cost, closure, and effort outcomes use paired cluster-bootstrap differences and include medians and tail quantiles. The target of 300 natural triggers provides power for an absolute SRR difference of approximately ten percentage points under moderate paired discordance; achieved power and all exclusions are recomputed from the frozen screening yield without replacing missing cases.

The main paper reports the end-to-end matched-budget SRR comparison, its principal safety outcomes, and the joint-necessity and three-generation summary in Figure 7. Appendix C.6 gives the full shared-proposal comparison; Appendix C.6 defines its metrics and accounting. This section reports every fault family, cell-status distribution, complete subcube denominator, safeguard ablation, human-audit breakdown, held-out-policy result, and token, execution, and wall-clock cost. Intermediate or inadmissible cells, promotion shards, and sealed audit cases are never added to the policy-training corpus.

B.6 Protocol Summary Tables

B.7 Canonical Trajectory Normalization and Diagnostics

Canonical and replayable trajectories. Harness adapters emit a canonical event schema that preserves policy messages, versioned tool definitions, tool calls and results, artifact references,

Table 17: Structural and procedural task descriptors used during construction; these descriptors do not by themselves define empirical difficulty.

Dimension	Operationalization and diagnostic evidence
State dependence	Later operations consume earlier artifacts or decisions; measured through dependency depth and recovery from upstream errors.
Cross-artifact coupling	Outcomes require consistent use of multiple files, formats, versions, or roots; input ablation tests whether each dependency is necessary.
Exact constraints	Numerical, textual, temporal, geometric, or formatting conditions are evaluated through criterion-level completion and violation counts.
Boundary constraints	Unrelated files, regions, styles, and historical state remain protected; unintended changes and protected-state mutations are detected.
Workspace navigation	Required evidence is embedded in versioned contexts with plausible distractors; diagnostics include discovery turns and wrong-version selection.
Observable side effects	Create, modify, delete, organize, or export operations are checked against the terminal diff and expected-output coverage.

Table 18: Outcome-anchor taxonomy used to instantiate verifiable rubric criteria.

Anchor	Observable target and evidence
Lexical	Unique text, labels, titles, or identifiers observed in artifact text or rendered content.
Numerical	Exact values, formulas, aggregates, and recomputable statistics observed in cells, tables, calculations, or tool outputs.
Structural	Named sheets, sections, layouts, ordering, or document organization recovered from parsed structure and metadata.
Artifact	File identity, format, location, existence, page count, or other file-level properties recorded in manifests and metadata.
Relational	Consistency across artifacts, versions, provenance links, and workflow stages, supported by joined evidence from multiple surfaces.

Table 19: Quality-control stages and reported quantities. Structural validity remains separate from outcome correctness.

Stage	Checks and reported quantities
Package checks	File readability, metadata completeness, path privacy, permissions, and initial-state restoration; report yield and failure categories.
Semantic review	Clarity, contextual sufficiency, execution consistency, openness, cross-file necessity, and verifiability; report first-pass rate, rewrites, and final discard rate.
Demonstrator execution	Feasibility under declared tools, state boundaries, and resource limits; report termination rate, turns, tool errors, and cost.
Executable validation	Starter/Reference contrast, mandatory-output removal, protected-state enforcement, and evidence coverage; report per-check success and unresolved cases.
Candidate-stage FCR	Contract-preserving repair of candidate-visible Phase II/III failures; report routed share, conditional recovery, final $\mathrm{SynthPass}_B$, intent drift, and matched-budget cost.
Human audit	Professional realism, internal coherence, and criterion–evidence agreement; report sample size, reviewer agreement, uncertainty, and difficulty breakdowns.
Dataset monitoring	Coverage over roles, workflows, artifacts, outputs, prompts, capabilities, and difficulty; report distributions and template concentration.

changed-file manifests, and workspace-state metadata. Every tool call must resolve to a matching result, every invoked tool must appear in the active registry, and a retained trace must either terminate normally or record a successful recovery from intermediate errors. Capabilities that are native to a particular model interface are exposed through declared Harness tools, and their returned evidence is materialized in the trace rather than left as an online reference. These invariants make execution evidence portable across Harness implementations and allow a rollout to be audited or replayed without the originating service. We additionally report behavioral diagnostics—planning-state continuity, dependency-aware parallel evidence collection, write–read-back coverage, error recovery, and redundant-call rate—as trace properties rather than outcome labels. Tables 20 and 21 specify the canonical record and diagnostics.

Different Harness implementations expose heterogeneous message fields and tool-event formats. Before training or evaluation, an adapter maps each trace to the canonical record in Table 20. The representation preserves original ordering and identifiers while separating policy-generated content, observations, tool events, and task metadata. Large tool registries are versioned and deduplicated; each rollout stores the registry identity and definitions exposed to the policy, avoiding repeated schema expansion while preserving reproducibility.

Table 20: Canonical trajectory record and integrity invariants.

Component	Canonical content and integrity invariant
Tool registry	Versioned function schemas and capability metadata; every invoked tool resolves to one active definition.
Policy messages	Ordered model responses, visible user-facing text, and tool-call requests; roles and turn boundaries are valid and untruncated.
Tool results	Result payload, call identifier, status, and materialized evidence; every call has a matching result and no dangling identifiers remain.
Artifact evidence	Referenced or produced files, renderings, and the changed-file manifest; evidence identifiers resolve and expected outputs are represented.
Progress evidence	Turn-boundary snapshots or replayable diffs, checker/schema hashes, evaluability flags, and versioned tool-output or artifact dependencies; each local score and propagation edge resolves to recorded evidence.
Task metadata	Context identity, Harness version, roots, capabilities, and difficulty descriptors; metadata agrees with the restored session.
Terminal record	Final policy response, termination status, and recovered intermediate errors; the trace terminates normally or documents successful recovery.

Table 21: Behavioral trajectory diagnostics. These measurements characterize execution quality but do not replace outcome scoring.

Diagnostic	Operational definition and failure pattern
Parallel evidence collection	Independent reads or lookups are grouped when dependencies permit; unnecessary serialization lowers coverage.
Write–read-back coverage	State-changing calls are followed by a read, list, render, or status check; missing checks indicate unverified side effects.
Error recovery	A failed call is diagnosed and followed by a corrected action; repeated failure or abandoned state is recorded.
Constraint preservation	Negative and boundary constraints remain satisfied after relevant writes; out-of-scope edits are violations.
Redundant-call rate	Repeated calls with equivalent normalized arguments and unchanged managed state reveal loops, empty retries, or repeated reads.
Successful closure	The final response identifies completed artifacts or conclusions supported by recorded evidence; truncation and unsupported claims fail this check.

B.8 Verifier-to-Turn Credit Assignment Details

This subsection expands the training procedure summarized in Section 3.3 and Figure 4. VTCA is a verifier-grounded progress-attribution rule implemented as workspace-grounded advantage shaping. Its dependency allocations describe recorded data use and do not estimate counterfactual causal effects.

Rollout groups and normalization. Canonical trajectory fields and validity checks follow Appendix B.7. The SFT loss is computed only on agent-generated tokens from teacher trajectories with Verifier status pass that also pass structural and lineage QA. For each task, we form a rollout group $\mathcal{G}_i = \{\tau_{i,r}\}_{r=1}^{M_i}$ of $M_i \geq 2$ evaluation-valid trajectories by sampling from the behavior policy $\pi_{\theta_{\text{old}}}$ with parameters held fixed within each group. Environment- or evaluator-invalid attempts are resampled within a fixed rollout budget; groups that remain undersized are omitted from the update and reported. Only the terminal reward is standardized within the group. For rollout r, the terminal Verifier supplies outcome score $S_{B,i,r} \in [0,1]$ and hard gate $G_{i,r} \in \{0,1\}$; frozen deterministic trace validation supplies process score $S_{A,i,r} \in [0,1]$. Their gated reward is

$$R_{i,r} = G_{i,r}\left[(1-\lambda)S_{B,i,r} + \lambda S_{A,i,r}\right], \qquad 0 \leq \lambda \leq 0.1. \tag{28}$$

The gate prevents graded scores from compensating for hard violations. Within each rollout group, we normalize this reward as

$$\bar{R}_i = \frac{1}{M_i}\sum_{r=1}^{M_i} R_{i,r}, \qquad A^{\mathrm{traj}}_{i,r} = \frac{R_{i,r} - \bar{R}_i}{\max(\mathrm{sd}_{r'}[R_{i,r'}], \sigma_{R,\min})}. \tag{29}$$

The floor $\sigma_{R,\min} > 0$ is fixed on development data. A zero-variance reward group has zero trajectory advantage but may retain local progress.

Progress checks and missing evidence. VTCA invokes the terminal Verifier once per trajectory, including at most one residual Judge call. Local checks use the frozen progress-check contract on recorded turn-boundary workspace and evidence states $x_{i,r,t}$, after all results of the preceding turn are captured. A checker returns $\phi_{i,k}(x) \in [0,1]$ only on evaluable states. Verified absence may be zero when its schema defines it as incomplete; unavailable observations, parser failure, and unsupported semantic criteria are missing values. The flag $v_{i,r,t,k} = 1$ requires both adjacent endpoints to be evaluable. For a rollout of $T_{i,r}$ turns, $x_{i,r,0}$ is initial and $x_{i,r,T_{i,r}}$ terminal; turn $t \in \{0, \dots, T_{i,r} - 1\}$ emits

$$\Delta_{i,r,t,k} = \begin{cases} \phi_{i,k}(x_{i,r,t+1}) - \phi_{i,k}(x_{i,r,t}), & \text{both endpoints evaluable,} \\ 0, & \text{otherwise (no local signal).} \end{cases} \tag{30}$$

Missing endpoints emit no local signal and retain their missing reason; unavailable observations are never treated as zero completion. Missing local evidence does not by itself invalidate a healthy terminal evaluation. Checker calls and state recording add cost, but no intermediate Judge calls.

Provenance-backed propagation. For task i, rollout r, and progress-producing turn s, let $\mathcal{P}_{i,r,s,k} \subseteq \{0, \dots, s-1\}$ contain prior turns whose tool outputs or artifact versions are consumed directly by the validated progress at turn s, with links consistent with D_{T_i} and the criterion's evidence binding. Mere surface access, entity mentions, and self-reported plans are insufficient; an unverified use receives no link. Deduplicate by originating turn, so multiple consumed outputs from one turn do not give it extra weight. Define $d_{i,r,s,t,k} = \mathbf{1}[t \in \mathcal{P}_{i,r,s,k}]/|\mathcal{P}_{i,r,s,k}|$ for a nonempty set, and zero otherwise. With $\rho_{i,r,s,k} = \rho \mathbf{1}[|\mathcal{P}_{i,r,s,k}| > 0]$, the fully indexed allocation and local signal are

$$\begin{aligned} b_{i,r,s,t,k} &= \begin{cases} 1 - \rho_{i,r,s,k}, & t = s, \\ \rho_{i,r,s,k}/|\mathcal{P}_{i,r,s,k}|, & t \in \mathcal{P}_{i,r,s,k}, \\ 0, & \text{otherwise,} \end{cases} \qquad 0 \le \rho \le \tfrac{1}{2}, \\ \delta_{i,r,t} &= \sum_{k=1}^{K_i} \bar{w}_{i,k} \left[\sum_{s=t}^{T_{i,r}-1} b_{i,r,s,t,k} \max(\Delta_{i,r,s,k}, 0) + \min(\Delta_{i,r,t,k}, 0) \right]. \end{aligned} \tag{31}$$

Thus $b_{i,r,s,t,k} = 0$ for $t > s$ and for unverified earlier recipients. The producing turn is included in the inner sum, and the negative term is specific to task, rollout, turn, and criterion. The coefficient ρ and the dependency rule are fixed on development data. Positive progress stays primarily at its producing turn; negative changes stay entirely there. Criterion weights remain $\bar{w}_{i,k} = w_{i,k}/\sum_j w_{i,j}$, with no renormalization over observed criteria or default centering across turns. The local term is gated by $H_{i,r}$ (integrity, protected state, and safety), while the terminal reward retains $G_{i,r}$, including critical completion. Thus useful steps in an unfinished but nonviolating trajectory remain eligible. The shaping interpretation and its limits are discussed below in Appendix B.8.

Provenance quality and allocation reliability. Recorded direct use is evidence for eligibility under the allocation rule; it does not establish that an earlier turn was necessary for the outcome. Incorrect producer or version matches can transfer shaping signal to an irrelevant turn. Missed dependencies leave prerequisites uncredited, and even exact matches can include inputs that the operation did not materially need. Thus provenance precision and recall limit allocation reliability and coverage. The conservative rejection of ambiguous links trades coverage for fewer unsupported allocations; it is not a guarantee of either metric. The sealed audit in Table 23 evaluates dependency precision/recall and signal localization separately from checker coverage. The bound on each propagated share limits its mass, not its semantic correctness.

Dependency extraction and allocation algorithm. The input is a completed, evaluation-valid canonical trace, its turn-boundary states, frozen criterion weights and checkers, and the versioned dependency contract. The output is $\delta_{i,r,t}$ and an auditable allocation ledger. The following procedure defines the direct-use rule; it does not recursively propagate credit through a transitive closure.

1. Index each tool output and artifact version by immutable identifier, producer turn, and content identity. Match calls to results; deduplicate replayed events by call ID. Initial artifacts have no credited agent turn.
2. At every turn boundary, run each supported checker. Record either a score with its evidence identifiers or a missing reason. Compute $\Delta_{i,r,s,k}$ only when both adjacent endpoints are evaluable; otherwise set the local increment to zero without bridging the missing interval.
3. For each positive increment, inspect the recorded inputs consumed by the operation producing the criterion's changed evidence. Resolve each input to its exact prior output or artifact version. Add a producer turn $t < s$ only when the use is explicit, consistent with D_{T_i}, and relevant to the criterion's evidence binding. A stale or ambiguous version, an inferred use, or an unrecorded dependency supplies no edge; retain its exclusion reason. Construction-only bindings remain in the reward computation, not the policy input.
4. Initialize every $\delta_{i,r,t}$ to zero. Allocate each weighted positive increment to its producer and the deduplicated set $\mathcal{P}_{i,r,s,k}$ using Equation 31. Add each weighted negative increment only to its producing turn. A turn may receive credit from several later increments; each source increment is allocated exactly once.
5. Store the source turn, criterion, increment, recipient IDs, allocation weights, and evidence links. Apply the trajectory-wide $H_{i,r}$ only when forming the shaped advantage, then subtract the local trace cost. Terminal normalization, all allocation values, and gates remain detached from gradients.

Conservation and boundary cases. For each source increment, $\sum_{t=0}^{T_{i,r}-1} b_{i,r,s,t,k} = 1$ and $b_{i,r,s,s,k} \geq 1/2$. Consequently, before gating, trace costs, clipping, or objective aggregation,

$$\sum_{t=0}^{T_{i,r}-1} \delta_{i,r,t} = \sum_{k=1}^{K_i} \bar{w}_{i,k} \sum_{s=0}^{T_{i,r}-1} \Delta_{i,r,s,k}.$$

This conserves the emitted increments, including their positive mass separately. It equals a terminal-minus-initial potential difference only when all required transitions are observed. Missing transitions emit no increment, and differences are not computed across missing intervals. For example, a single criterion's $+0.8$ increment at turn 2, with prerequisites 0 and 1 and $\rho = 0.25$, allocates $(0.1, 0.1, 0.6)$; a later -0.2 regression remains at its own turn. This arithmetic example illustrates credit conservation. Required implementation checks cover empty and multiple prerequisite sets, duplicate edges, delayed direct use, stale versions, missing endpoints, $\rho = 0$, $\rho = 1/2$, and $H = 0$. An unfinished but nonviolating trajectory may have $G = 0, H = 1$ and retain local progress. Write–revert cycles must preserve the emitted signed mass, but that property does not guarantee an unchanged clipped policy update or immunity to reward manipulation.

Shaping interpretation and theoretical limits. Potential-based reward shaping adds $\gamma\Phi(x') - \Phi(x)$ to the environment reward and preserves optimal policies under its assumptions (Ng et al., 1999). RUDDER studies return-equivalent reward redistribution using learned return decompositions (Arjona-Medina et al., 2019). VTCA instead adds a workspace-progress term to a group-relative trajectory advantage. The conservation identity above concerns emitted checker increments; it does not make the shaped signal a decomposition of the original episode return.

The allocation can depend on later recorded uses of an artifact, and its effect on the update also depends on gating, trace costs, clipping, and turn/token aggregation. Consequently, conservation alone establishes neither policy invariance nor an unbiased policy gradient. A write–revert cycle can have zero summed increments while still changing the clipped update. Likewise, a checker increase is evidence of progress on its declared criterion, not an estimate of a turn's causal contribution to final success. The empirical controls compare the usefulness of this shaping signal and its recipients under fixed budgets and observation rules.

Turn-level trace cost. The deterministic cost $c_{i,r,t} \in [0, 1]$ is nonzero for malformed tool calls, explicit tool errors, repeated actions that leave the state unchanged, and trace-attributable protected-state or safety violations. The cost is assigned only to the turn at which the event occurs. It is distinct from the trajectory-level process score S_A and has its own coefficient η in Equation 1; neither signal uses Judge output.

Turn-balanced GRPO objective. For agent-generated token $a_{i,r,t,\ell}$ at position ℓ of turn t, conditioned on prefix $\xi_{i,r,t,\ell}$, define the importance-sampling ratio, clipped surrogate contribution, and

exact forward KL divergence between the full next-token distributions as

$$\begin{aligned} \varrho_{i,r,t,\ell}(\theta) &= \frac{\pi_\theta(a_{i,r,t,\ell} \mid \xi_{i,r,t,\ell})}{\pi_{\theta_{\text{old}}}(a_{i,r,t,\ell} \mid \xi_{i,r,t,\ell})}, \\ \psi^{\text{clip}}_{i,r,t,\ell} &= \min\Big(\varrho_{i,r,t,\ell}\widehat{A}_{i,r,t}, \text{clip}(\varrho_{i,r,t,\ell}, 1-\varepsilon_c, 1+\varepsilon_c)\widehat{A}_{i,r,t}\Big), \\ D^{\text{KL}}_{i,r,t,\ell} &= D_{\text{KL}}(\pi_\theta(\cdot \mid \xi_{i,r,t,\ell}) \,\|\, \pi_{\text{ref}}(\cdot \mid \xi_{i,r,t,\ell})). \end{aligned} \tag{32}$$

Here $\pi_{\theta_{\text{old}}}$ is the behavior policy held fixed during an update, and π_{ref} is the frozen reference policy for the corresponding RL run (the SFT policy for VTCA). Let $L_{i,r,t} \geq 1$ denote the number of agent-generated tokens in turn t. We maximize

$$\mathcal{J}_{\text{VTCA}}(\theta) = \mathbb{E}_{\substack{i\sim\mathcal{D}_{\text{train}},\\ \mathcal{G}_i\sim\pi_{\theta_{\text{old}}}}}\left[\frac{1}{M_i}\sum_{r=1}^{M_i}\frac{1}{T_{i,r}}\sum_{t=0}^{T_{i,r}-1}\frac{1}{L_{i,r,t}}\sum_{\ell=1}^{L_{i,r,t}}\left(\psi^{\text{clip}}_{i,r,t,\ell} - \beta D^{\text{KL}}_{i,r,t,\ell}\right)\right]. \tag{33}$$

The nested reduction first averages token-level contributions within each turn, then turns within each trajectory, and finally trajectories within each rollout group. This prevents long turns from dominating the gradient estimate. Rewards, checker outputs, dependency allocations, normalization statistics, and advantage estimates are detached from the computation graph. We report M_i, resampling rules, α, η, β, ρ, the reward standard-deviation floor, clipping bounds, and optimizer hyperparameters for each configuration.

Progress, propagation, and aggregation controls. Table 22 specifies the component controls needed to interpret the aggregate comparison in the main text. The eight core conditions distinguish local progress, recipient identity, trace cost, and aggregation. Every row uses the same SFT initialization and task–seed schedule. The primary comparison fixes attempted-rollout and terminal-evaluation caps; Table 24 reports realized resources, including local checks and state capture. A companion comparison fixes total compute using predeclared CPU/GPU accounting and charges these costs. Equal rollout caps alone do not establish equal compute. Unless specified, rows use the turn-balanced objective in Equation 33.

Conditions A and B are terminal-only baselines with different aggregation; C adds local progress with $\rho = \eta = 0$; D redistributes that progress to random earlier turns with $\eta = 0$; E uses verified dependencies with $\eta = 0$; F adds the trace cost to E; G uses F with token-weighted aggregation; and H adds only trace cost to B. All rows share the terminal reward, including λ, the reference policy, KL coefficient, clipping bounds, and validity rules. The primary mechanism contrasts are E–C and E–D. C–B estimates the effect of local progress; F–E and H–B test trace cost; B–A and F–G test aggregation. The additional $\rho = 0$ control with trace cost retained tests whether the propagation effect depends on that cost.

For D, each positive event retains the same producing-turn mass, propagation budget, and number of distinct earlier recipients as E. When $m = |\mathcal{P}_{i,r,s,k}| > 0$, sample m distinct turns uniformly from $\{0, \ldots, s-1\}$ using a logged task–rollout–event seed and allocate ρ/m to each. Do not exclude true prerequisites: report their chance overlap. Empty prerequisite sets stay local. Thus the randomized control changes recipient identities without changing the eligible-event mask or credit budget.

The default signal is $\widetilde{\delta}_t = \delta_t$. Centering controls use either $\widetilde{\delta}_t = \delta_t - T^{-1}\sum_s \delta_s$ or support-only centering. For the latter, $e_t = 1$ iff turn t produces a nonzero valid increment or receives a positive allocation before cancellation; suppressing (i, r),

$$\bar{\delta}^{\text{supp}} = \frac{\sum_t e_t\delta_t}{\max(1, \sum_t e_t)}, \qquad \widetilde{\delta}^{\text{supp}}_t = e_t(\delta_t - \bar{\delta}^{\text{supp}}). \tag{34}$$

Centering is an ablation, not a conservation or invariance requirement. We report the mean absolute local term $\mathbb{E}|\ell_t|$, where $\ell_t = H\widetilde{\delta}_t$ by default and uses the specified gate in gate controls.

Hyperparameter and coverage sensitivity. The sensitivity protocol freezes a disjoint development cohort and three development training seeds before examining results. Change one coefficient at a time: α and η use $\{0, 0.5, 1, 2\}$ times their recorded positive defaults, $\rho \in \{0, 0.25, 0.5\}$, and $\lambda \in \{0, 0.05, 0.1\}$. Absolute defaults and all resulting values must be reported; a zero default does not define a nonzero multiplier sweep and must be identified as an inactive component. Terminal λ changes apply to both B and F in paired comparisons. Gate controls change only the local gate from H to G or one, preserving the terminal reward and official status. These are development controls, not test-set model selection.

To test checker availability, deterministically retain 25%, 50%, 75%, or 100% of supported criterion IDs under nested, seeded masks. Keep all terminal criteria and their original weights; do not

Table 22: VTCA component and aggregation controls under fixed attempted-rollout and terminal-evaluation caps. Pass denotes the WORKROVER test endpoint; external-suite results are recorded separately for each condition and seed.

Variant	Pass↑	$\mathrm{AUC}_{\mathrm{att/call}}$↑	$\mathbf{Pr(G=0)}\downarrow$	$\bar{\mathbf{c}}\downarrow$	$\mathbf{E}\lvert\ell_t\rvert$
Credit components and aggregation					
A: Terminal only; token-weighted	52.6	0.484/0.484	0.220	0.082	0.0000
B: Terminal only; turn-balanced	54.0	0.494/0.494	0.200	0.078	0.0000
C: Local progress; $\rho = \eta = 0$	55.8	0.510/0.510	0.190	0.076	0.0048
D: Random recipients; $\eta = 0$	55.6	0.508/0.508	0.192	0.077	0.0048
E: Provenance recipients; $\eta = 0$	57.8	0.527/0.527	0.180	0.074	0.0048
F: Full VTCA; turn-balanced	59.0	0.539/0.539	0.160	0.048	0.0047
G: Full VTCA; token-weighted	57.6	0.525/0.525	0.175	0.054	0.0046
H: Terminal + trace cost; $\alpha = 0$	55.2	0.504/0.504	0.180	0.052	0.0000
Centering, dependency, and gating controls (full VTCA; turn-balanced)					
All-turn centering	55.9	0.510/0.510	0.188	0.055	0.0075
Support-only centering	57.9	0.528/0.528	0.170	0.051	0.0042
Cardinality-matched shuffled dependencies	57.0	0.519/0.519	0.180	0.050	0.0047
No propagation; trace cost retained	57.0	0.519/0.519	0.180	0.050	0.0047
Terminal gate G for local shaping	57.0	0.519/0.519	0.170	0.050	0.0034
Ungated local shaping	58.3	0.532/0.532	0.205	0.052	0.0052
No process-score term ($\lambda = 0$)	58.8	0.538/0.538	0.164	0.051	0.0047
Companion outcomes at matched total compute (turn-balanced)					
Episode-only GRPO	54.0	0.494/0.494	0.200	0.078	0.0000
Full VTCA	58.2	0.533/0.533	0.166	0.050	0.0047
Same-task rollout-group size					
$M_i = 2$	57.8	0.526/0.526	0.175	0.052	0.0046
$M_i = 4$	59.0	0.539/0.539	0.160	0.048	0.0047
$M_i = 8$	58.6	0.535/0.535	0.163	0.048	0.0048

Episode-only GRPO sets $\alpha = \eta = 0$. Shuffled dependencies preserve the number of earlier recipients and propagated mass, but randomize their identities. Gate controls replace only H in local shaping by G or one; the terminal reward and official task status remain unchanged. $\bar{c}$ is the mean turn-level cost on valid trajectories. $\mathrm{AUC}_{\mathrm{att/call}}$ is indexed by attempted rollouts / terminal Verifier calls.

renormalize the remaining local signals. Report the realized supported weight, evaluable-pair coverage, hard-gated pass rate, tool cost, and seed dispersion for every setting. The full-data default and confirmatory contrasts are fixed before testing.

Sealed credit-localization audit. Two annotators, blinded to method, local scores, and ablation identity, label progress-producing turns, valid prerequisite links, and regression-producing turns from the frozen rubric, trace, and recorded state changes. Adjudication, sample size, agreement, and the predeclared k are reported; labels never enter training or model selection. Table 23 measures dependency precision/recall, responsible-turn recall, and signal outside human-supported turns. For label $u_{i,r,t} \in \{0, 1\}$ and signal $s_{i,r,t}$, define

$$\mathrm{Off}(s) = \frac{\sum_t (1 - u_{i,r,t}) |s_{i,r,t}|}{\epsilon + \sum_t |s_{i,r,t}|}. \tag{35}$$

We report $\mathrm{Off}(\delta)$ and $\mathrm{Off}(\ell)$ on nonzero signals with their trajectory coverage, separating positive and negative events. Semantic criteria without a progress checker have no local signal and are reported as unsupported, rather than as failed localization. A separate checker audit reports supported criterion weight, evaluable endpoint-pair coverage, missing reasons, and signed-change agreement with independently checked state changes. Write–revert cycles, repeated progress, stale artifact versions, unsupported evidence links, and policy-induced unobservability (e.g., corrupting a parser input to hide a regression) are adversarial cases. Human relevance and provenance agreement do not establish causal contribution.

The audit unit is a task–rollout–criterion event, with Work Context retained for clustered inference. Sample independently of recorded dependency links so missed links enter the recall denominator; include no-progress and unsupported cases in the sampling ledger. Two annotators independently label exact source and consumed version IDs, progress signs, and supporting turns; a third adjudicates disagreements without viewing method scores. Report pre-adjudication agreement, adjudicated counts,

exclusions, and unresolved labels by horizon and evidence type. The default responsible-turn summary uses $k = 3$, with $k = 1, 5$ as secondary summaries and ties resolved by turn ID.

Table 23: Sealed dependency and localization audit on supported criteria. Brackets report nonzero-signal coverage; checker applicability and missing evidence are reported separately.

Local shaping	Dependency P↑	Dependency R↑	Responsible-turn R@k↑	Off(δ) ↓	Off(ℓ) ↓ [coverage]
Full VTCA	0.94	0.78	0.74	0.07	0.06 [0.86]
Without propagation ($\rho = 0$)	–	–	0.50	0.02	0.02 [0.82]
All-turn centering	0.94	0.78	0.71	0.07	0.34 [0.86]
Support-only centering	0.94	0.78	0.72	0.07	0.10 [0.86]
Shuffled dependencies	0.12	0.10	0.52	0.23	0.21 [0.86]

Realized budgets, gate sparsity, and signal overlap. Table 24 records realized rollout, optimizer, terminal-evaluation, and progress-check resources against frozen caps. Compute includes local checker CPU time and state capture, with wall time and recorded bytes reported separately. The total-compute companion comparison uses the same accounting for every method. Signal diagnostics separate all-$G = 0$ from all-$H = 0$ groups, zero trajectory advantages, and missing or zero local signals, stratified by horizon, tool family, and Work Context. The final panel measures overlap among S_A, c_t, and the violation gate H so that repeated penalties are visible. Supported weight is the mean total criterion weight with calibrated checkers; valid-pair coverage is the fraction of turn–criterion pairs with $v = 1$ among supported criteria. The $\delta = 0$ rate counts valid trajectories with no nonzero local signal, with missing-only cases separated.

Realized backward-credit mass. Let $a_{i,r,s,k} = \bar{w}_{i,k} \max(\Delta_{i,r,s,k}, 0)$ and let sums below run over all valid task–rollout–source–criterion events. Report

$$q_{\text{event}} = \frac{\sum \mathbf{1}[a > 0,\ |\mathcal{P}| > 0]}{\sum \mathbf{1}[a > 0]}, \qquad q_{\text{mass}} = \frac{\sum a \rho_{i,r,s,k}}{\sum a}.$$

Compute the same two quantities on $H = 1$ trajectories to describe the signal that reaches the optimizer. Report numerator and denominator as well as the ratio; a zero denominator is unavailable, not zero propagation. With a fixed ρ, $q_{\text{mass}} \le \rho \le 1/2$, whereas q_{event} need not be small. Also report the number of recipients and backward distance $s - t$, weighted by allocated mass. These measurements distinguish a configured propagation coefficient from how often verified dependencies actually occur.

Table 24: Realized VTCA resource ledger, signal-degeneracy diagnostics, and overlap among process penalties. Both budget caps and realized consumption are reported; no unobserved resource is treated as matched by assumption.

(a) Realized budget ledger

Method	Attempted	Valid	Agent tokens	Updates	Verifier	Judge	Compute
Episode-only; token-weighted	102400	98304	3276800000	400	102400	58982	1.04
Episode-only; turn-balanced	102400	98816	3174400000	400	102400	59290	1.00
Full VTCA; token-weighted	102400	98816	3072000000	400	102400	59290	1.12
Full VTCA; turn-balanced	102400	99328	2969600000	400	102400	59597	1.08
Episode-only; total-compute cap	102400	98816	3174400000	400	102400	59290	1.00
Full VTCA; total-compute cap	94720	91878	2746880000	370	94720	55127	1.00

(b) Rollout validity and signal degeneracy (%)

Method	Yield↑	Invalid↓	Skipped↓	All-$G = 0$↓	All-$H = 0$↓	$\mathbf{sd}(\mathbf{R}) < \sigma_{\mathbf{R}}$	$\mathbf{A}^{\text{traj}} = 0$
SFT + Episode-only GRPO	96.5	3.5	1.0	12.0	0.3	17.5	14.8
SFT + VTCA-GRPO	97.0	3.0	1.0	8.0	0.2	11.0	9.0

(c) Local-check cost and coverage

Method	Check calls	CPU time	Check/capture wall	State bytes	Supported weight	Valid pairs	$\delta = 0$
Episode-only; turn-balanced	0	0	0/0.12	3145728000	–	–	–
Full VTCA; turn-balanced	61440000	122880	1.07/1.78	41943040000	0.65	0.92	0.18
Full VTCA; total-compute cap	56832000	113664	0.99/1.64	38797312000	0.65	0.92	0.19

(d) Trace-derived signal overlap (fraction of triggered events)

Event family	Affects S_A	Affects c_t	Triggers $H = 0$	Any overlap↓
Wrong tool/parameter or unrecovered error	1.00	1.00	0.02	1.00
Repeated unchanged-state action	0.85	1.00	0.00	0.85
Protected-state or safety violation	1.00	1.00	1.00	1.00

Independent Verifier and score robustness. An independently implemented executable Verifier and an alternate Judge rescore sealed terminal bundles while blind to method and official scores. Independent deterministic checks additionally inspect recorded intermediate states for supported criteria. We report terminal gate disagreement, criterion rank agreement, signed-progress agreement, and independently confirmed pass rate; terminal rescoring alone does not validate local credit. Non-gate rubric score perturbations test trajectory-reward sensitivity while all gates and progress-check contracts remain fixed. All audits are held out from training and model selection. Periodic curves use development data only; the final model is evaluated once on the test set after fixed-budget training.

Table 25: Independent terminal and progress-check audits and non-gate score sensitivity. Dashes denote conditions without independent local checking.

Condition	Gate disagree↓	$\rho(\mathbf{p}, \mathbf{p}')$ ↑	Progress-sign agree↑	Pass↑	$\mathrm{AUC}_{\mathrm{att}}$↑
Locked official Verifier	0	1	1	59.0	0.539
Independent Verifier and local checks	0.012	0.970	0.960	57.8	0.529
Alternate blinded Judge	0.008	0.940	–	58.4	0.534
Non-gate rubric perturbation: 5%	–	0.980	–	58.8	0.538
Non-gate rubric perturbation: 10%	–	0.950	–	58.4	0.535

C Additional Experiments and Evaluation Details

Our experiments address six questions. *(RQ1)* Can WorkRover synthesize diverse, professionally coherent training environments at scale? *(RQ2)* Are the resulting tasks executable and their state-grounded Verifiers reliable under counterfactual testing? *(RQ3)* How well do current agents perform as workspace scale, workflow horizon, and dependency complexity increase? *(RQ4)* Does VTCA improve over episode-only RL under matched budgets, and does its turn-level signal track verifiable progress and prerequisite actions? *(RQ5)* Does candidate-stage FCR increase end-to-end task-package synthesis pass rate, and does maintenance-stage FCR resolve confirmed construction mismatches more reliably than localized, sequential, or whole-package repair under matched budgets? *(RQ6)* Which component interactions account for performance differences, and which closure, anchoring, and promotion safeguards prevent spurious improvements?

C.1 Experimental Setup

Evidence status and reporting scope. The primary VTCA results in Figure 6 report the matched-rollout-budget comparison of SFT + VTCA-GRPO and SFT + Episode-only GRPO on all five external benchmarks, as described in Section 4.3.1. This experiment is separate from the published reference-model scores in Table 1. Numerical benchmark entries are reported aggregate scores. In particular, aggregate scores alone do not establish five-seed reproducibility, the effect of an individual VTCA component, or criterion-specific Verifier reliability. These distinctions apply to the supplementary controls in Appendix B.8 as well.

Models and training data. We post-train only Qwen3.5-35B-A3B. In the matched learning study, Backbone denotes this checkpoint before our SFT or RL. We compare Backbone, SFT, Backbone + Episode-only GRPO, SFT + Episode-only GRPO, and SFT + VTCA-GRPO on the 35B-A3B model; workspace- and Harness-scaling, recursive environment evolution, and rejection-sampling studies use the same model. SFT uses 30,115 Verifier-accepted trajectories that also pass structural and lineage QA. Controlled workspace or Harness comparisons allocate training examples equally across the varied dimension, and we independently synthesize **20000** candidate task–Verifier pairs for RL. We split data at the Work Context level, keeping every task derived from the same file-system state in one split; evaluation includes in-distribution, held-out-role, and held-out-artifact-composition sets.

Training details. The model identities reported in Section 4.1 are Qwen3.5-397B-A17B for SFT demonstrations and Qwen3.5-35B-A3B for pilot screening; the SFT-initialized 35B-A3B student collects online RL rollouts. The collection protocol uses prefix-preserving rejection sampling at agent-turn boundaries, allowing at most **3** retries for a rejected turn while retaining the verified prefix and its exact workspace state. For RL-data selection, the pilot model executes **4** rollouts for each candidate task–Verifier pair; consistency filtering retains **3,200** pairs from the initial pool of **20000**. The 35B-A3B RL runs start from the corresponding SFT model and use the turn-balanced GRPO objective in Section 3.3, with a KL coefficient of **0.01**. Episode-only GRPO sets $\alpha = \eta = 0$ in Equation 1. VTCA additionally records intermediate state and evidence for frozen deterministic

progress checks. All conditions share the same system interface, tools, permissions, initial snapshot, and rollout budget; a companion comparison matches total compute including checks and state capture.

Training population and visualization subset. The visualization subset contains 3,200 selected traces from a parent collection of 5,000 successful traces. Selection requires $40 <$ deduplicated assistant turns < 200, `eval.score`> 0.6, `eval.passed=true`, complete segments, and completed termination. Candidates are ordered first by the presence of a named skill call, then by evaluation score, with a deterministic hash tie-breaker. These statistics describe a length-restricted, success-filtered, skill-prioritized subset; they do not estimate the unfiltered online RL distribution or establish that the same filtering is used to choose optimizer updates. The threshold 0.6 is a visualization-subset selection threshold and is not evidence of a universal task acceptance threshold τ_B. Failed but evaluation-valid RL rollouts remain eligible under the training definition.

Figure 5(d) presents the Harness allocation: OpenCode 60%, Claude Code 20%, Codex 10%, and Hermes 10%. Panels (a–c) summarize the selected 3,200-trace visualization subset described above; panel (d) reports the Harness allocation separately. Collector checkpoint and run identifiers are needed to attribute this subset to a particular training run.

Table 26: Training and collection populations.

Population	Count	Interpretation
SFT demonstrations	30,115	Reported accepted and QA-filtered training set.
Successful parent traces	5,000	Parent of the visualization subset.
Selected traces	3,200	Restricted subset described above.
Initial RL task–Verifier pairs	20000	Before pilot screening.
Retained RL task–Verifier pairs	3,200	After screening; distinct task count.
Attempted / valid / update-used RL traces	102400/ 99328/ 98304	Separate counts per run, including exclusions.
Completed independent training seeds	5	Run IDs and checkpoints required.

Run and evaluation manifest. Each result requires the model checkpoint, code and data revisions, Work Context split, task IDs, training and decoding seeds, optimizer settings, and attempted rollout, token, update, and wall-clock caps. The evaluation record additionally contains benchmark release and split, evaluator and Judge versions, Harness configuration, per-task outputs and scores, retry limits, invalid reasons, and the number of independent tasks and repeated attempts. Failed, invalid, resampled, and skipped attempts remain visible in the resource ledger. Realized resources are reported separately from caps; matching one budget does not imply matching all others.

Benchmarks. We evaluate Qwen3.5-35B-A3B and its post-trained variants on five external benchmarks: JobBench (Li et al., 2026g) for occupational tasks with expert-authored rubrics; GDPval (Artificial Analysis, 2026c) for professional deliverable quality; Automation-Bench (Shepard & Salimans, 2026) for cross-application business workflows; the Office subset of WorkBuddy Bench (Tencent WorkBuddy Bench Team, 2026a) for office data and mixed-format file workflows; and OfficeQA Pro (Opsahl-Ong et al., 2026) for grounded reasoning over U.S. Treasury Bulletins. The controlled OfficeQA Pro protocol uses the 133-question Pro split and accuracy (%) at a 0.0% numerical error tolerance. Additional reported runs retain their recorded configurations below. For our controlled runs, we use each benchmark's official evaluator and aggregate metric, with the dataset split, evaluator version, Harness, and rollout budget fixed across compared methods within each suite. Results are reported separately in Table 1: GDPval uses Elo ratings from pairwise comparisons, while the other suites retain their benchmark-defined scores. We do not average these different score scales. These five benchmark scores are the primary endpoints for the transfer comparison; the matched-budget VTCA study reports the same five suites in Figure 6. Supplementary WORKROVER diagnostics use the official hard-gated pass rate in Equation 11. Invalid evaluation attempts are not counted as task failures and may be rerun only up to a development-frozen cap, with every attempt charged to the same budget. Stochastic comparisons use repeated rollouts from identical snapshots, average rollouts within task, and cluster tasks by Work Context. We report paired differences with 95% cluster-bootstrap confidence intervals and additionally resample training seeds for trained-policy comparisons.

Published baseline selection. The reference-model roster follows Table 1 and Section 4.1. It builds on NexForge's Table 3 and Figure 1 (Zhao et al., 2026), with updated GLM-5.2 and MiniMax-M3 baselines (Z.ai, 2026; MiniMax, 2026; Artificial Analysis, 2026d;f). It also includes DeepSeek-

V4-Pro (DeepSeek-AI, 2026), Kimi K3 (Moonshot AI, 2026; Artificial Analysis, 2026e), Nex-N2-mini (Nex AGI, 2026), GPT-5.5 (OpenAI, 2026), Claude Opus 4.7 (Anthropic, 2026b), and Gemini 3.1 Pro Preview (Google DeepMind, 2026). The comparable-scale roster includes Qwen3.6-27B (Qwen Team, 2026b), Granite-4.2-30B, Muse-Glimmer-30B (IBM Granite Team, 2026; Meta Models, 2026), and Occamy-1.0 (35B total, 3B active parameters) (Accio-Lab, 2026). DeepSeek-V4-Pro denotes the August 13 official release (0813). Qwen3.5-35B-A3B (Qwen Team, 2026a) is the sole training backbone. The following notes document the sources and evaluation settings of the reported scores used in Table 1 and the cross-model comparisons in Section 4.2.

Occamy-1.0 reference results. The model card reports 27.60 on AutomationBench (Pass@1) and 48.10 on OfficeQA Pro (Accio-Lab, 2026), accessed September 16, 2026. The technical report specifies AutomationBench v1.0.6 with 600 public tasks and strict all-assertions pass scoring, and the 133-task OfficeQA Pro split with one attempt per task and exact-match scoring (Accio Team, 2026). Its GDPval score of 1128 uses an independently selected 207-task cohort from the public release and a different judge panel; the report explicitly distinguishes this evaluation from Artificial Analysis's v2 protocol. We use 1128 as Occamy-1.0's reported GDPval score in Table 1. For JobBench and WorkBuddy (Office), we use the supplied scores of 34.2667% and 0.6665, respectively. The table rounds JobBench to 34.27% and converts WorkBuddy (Office) to 66.65% to match its percentage scale.

Cross-model rankings. The cross-model comparison follows Section 4.2: rankings and score margins use the reported entries in Table 1, within its model groups and benchmark columns. Among the listed 27–35B open-weight models, WORKROVER-35B-A3B has the highest reported score on all five benchmarks, and Occamy-1.0 has the second-highest reported score on each. Compared with the listed models above 400B, WORKROVER exceeds GLM-5.2 and MiniMax-M3 on AutomationBench and OfficeQA Pro, and Kimi K3 on WorkBuddy Bench (Office), while its GDPval score remains below all models in that group. These comparisons retain each source's task subset, evaluator, scoring definition, and Harness, as documented in the benchmark-specific notes. Figure 1 visualizes the corresponding scores for its selected models.

Matched training comparison. The controlled comparison between SFT + VTCA and SFT + Episode-only GRPO is reported in Figure 6. The SFT + VTCA scores in Figure 6 match the corresponding entries in Table 1: 36.89 on JobBench and 28.67 on AutomationBench. The corresponding SFT + Episode-only GRPO scores are 33.5 and 23.0. These AutomationBench results are included in the primary five-benchmark matched comparison. The paired runs share the benchmark release, split, denominator, and strict-pass implementation.

GDPval sources and snapshot. Apart from Occamy-1.0's independent evaluation described above, the published GDPval references in Table 1 and Figure 1 use a single snapshot accessed on September 12, 2026 (Asia/Shanghai). Artificial Analysis's independent GDPval v2 uses Stirrup (Artificial Analysis, 2026c); the selected configurations are GPT-5.5 xhigh, Opus 4.7 adaptive/max, DeepSeek-V4-Pro max, GLM-5.2 max, reasoning-enabled MiniMax-M3, Kimi K3 max, Qwen3.6-27B Reasoning, Granite-4.2-30B, Muse Glimmer high, and the listed Gemini release. DeepSeek uses the August 13 model record (max reasoning effort), with 1493.31 Elo in the same September 12 snapshot. Table values retain the snapshot's Elo to two decimal places; Figure 1 rounds labels to integers while using the precise values for bar heights. Trailing zeros do not imply additional source precision. Earlier leaderboard snapshots are retained in the source archive but are not mixed into the selected reference column. The authors confirm that the WORKROVER result of 1218.00 Elo and all GDPval scores in Figure 6 use GDPval v2. The Qwen3.5-35B-A3B entry in Table 1 uses the same author-reported v2 score of 804.75 as Figure 6. It replaces the external score of 813.00 from NexForge's internal NexAU v1 reproduction, which anchors GPT-5.1 at 1000 (Zhao et al., 2026). Nex-N2-mini's 1097.03 is retained from the supplied record. Author-reported scores are not presented as independently verified entries on Artificial Analysis's public leaderboard.

Published JobBench protocols. The current JobBench-Main leaderboard uses 65 tasks, OpenCode v1.14.18, and the Grok 4.3 judge (JobBench Team, 2026). Its Gemini entry is named Gemini 3.1 Pro; the API endpoint revision is not disclosed. †Qwen3.6-27B uses the self-reported JobBench score of 21.8 from the official Qwen3.8-27B model card (Qwen Team, 2026c); the card does not specify the JobBench split, agent Harness, or judge configuration.

Published AutomationBench and Office protocols. AutomationBench references were checked on September 12, 2026 (Asia/Shanghai), prioritizing the 600-task Public set and its strict all-assertions pass rate (%). The official repository reports Kimi K3 (max) at 46.67 and GLM-5.2 (max) at 26.17 (Zapier, 2026). Moonshot's Public comparison reports GPT-5.5 (xhigh) at 22.70 (Moon-

shot AI, 2026). DeepSeek's model card reports 31.80 on AutomationBench Public for DeepSeek-V4-Pro-0813; we use this August 13 release throughout (DeepSeek-AI, 2026). Qwen3.6-27B's 9.67 is an independent reproduction on Public v1.0.5 (58/600 tasks, API toolset, no reasoning-effort setting), not an official Qwen result (Hert4, 2026). Nex-N2-mini's supplied 5.70 is also reported as Pass@1 in Accio-Lab's comparison; the inspected card does not explicitly identify its AutomationBench dataset revision (Accio-Lab, 2026). Evaluator versions and Harnesses vary across these Public records; Table 1 retains their reported scores.

For Gemini 3.1 Pro Preview (37.54), MiniMax-M3 (15.90), and Muse-Glimmer-30B (6.83), Table 1 uses the reported AutomationBench-AA scores. AA measures average objective completion on 657 private tasks, with zero for guardrail violations or execution errors, rather than strict task pass rate (Artificial Analysis, 2026a;b). Claude Opus 4.7 (9.90) retains the original paper's private-set strict pass rate (Shepard & Salimans, 2026). Author-supplied AutomationBench scores are 0.02% for Granite-4.2-30B and 6.00% for Qwen3.5-35B-A3B. The AutomationBench column preserves each source's dataset and scoring definition. [j]WorkBuddy Office uses the original paper's 50-task subset, think mode, and Claude Code, averaged over three runs (Tencent WorkBuddy Bench Team, 2026a). [l]Opus 4.7 uses the historical Office v1.2 leaderboard score with CodeBuddy Code in think mode; that published row does not document the task count, run date, or evaluator details (Tencent WorkBuddy Bench Team, 2026b). Online reference scores were accessed September 6, 2026, with GLM-5.2, MiniMax-M3, and the Kimi K3 references updated on September 7, 2026. The Qwen3.6-27B, Granite-4.2-30B, and Muse-Glimmer-30B records were added from the live leaderboards on September 9, 2026. The main-text comparisons use these source-specific scores and configurations.

Published OfficeQA Pro protocols. [m]Databricks' official OfficeQA Pro chart reports end-to-end accuracy on the 133-question Pro split over the full Treasury Bulletin PDF corpus: GPT-5.5 uses Codex SDK, Opus 4.7 uses Claude Agent SDK, and Gemini 3.1 Pro Preview uses Antigravity CLI (Databricks AI Research, 2026). We use the updated Antigravity result for Gemini, identified by an asterisk in the source chart. The repository dates the GPT-5.5, Opus 4.7, and Gemini updates to July 20, April 21, and August 2, 2026, respectively; it does not restate per-run error tolerances or rollout budgets for these updated scores. [n]Moonshot reports Kimi K3 and GLM-5.2 at max reasoning effort with Claude Code, providing the entire PDF corpus rendered as images without machine-readable text (Moonshot AI, 2026). [o]MiniMax reports M3 with Claude Code and exact-match scoring, supplying relevant files through a file system; the release does not specify full-corpus versus selected-file access or the document representation (MiniMax, 2026). These source-specific settings are retained as published references. OfficeQA Full and Pro V2 scores are not substituted for Pro scores. The published scores were accessed September 8, 2026.

Additional OfficeQA Pro runs. [p]WORKROVER-35B-A3B uses Claude Code with temperature 0.7, top-p 0.95, and top-k 40, with a 1,800-second timeout. The supplied evaluation record reports 48.87% without specifying the evaluator version or numerical error tolerance.

C.2 Scale and Quality of Environment Synthesis

Construction resources. The skill inventory contains 21,444 app skills, 5,918 file skills, and 36,064 job skills covering 317 roles (Appendix B.1.1). App and job skills each receive one secondary-category label, covering 15 and 10 categories, respectively. App skills are concentrated in data and AI platforms (21.9%), development and cloud platforms (21.4%), and content, design, and media (7.9%). The largest job categories are engineering (39.0%), project and quality management (13.9%), and data and AI (13.0%). File skills can receive multiple labels, producing 11,273 category memberships across 22 categories. The largest membership shares are text/Markdown (20.8%), structured data (14.0%), code, configuration, and repositories (11.6%), generic files and folders (9.3%), image/design (7.8%), and PDF (6.0%). Figure 14 shows the six largest named categories in each library, with all remaining categories combined as Other.

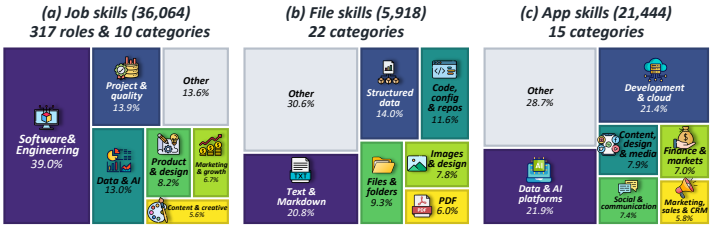

Figure 14: **Skill-library coverage.** Six largest categories; others aggregated. Tile sizes show shares of skills (a, c) or category memberships (b).

Scale, coverage, and construction yield. To evaluate scalability, we report the sizes of the raw and curated base pools, the numbers of candidate and accepted Work Contexts, and the number of tasks instantiated per context. Coverage is measured over occupational roles, job skills, file skills, artifact categories, and workflow families. We further characterize each file-system state by its files, directory roots, relation edges, cross-root dependencies, contextual-noise ratio, and relation-type distribution. Construction yield is reported at each filtering stage, separating the first-attempt rate, the share routed to candidate-stage FCR, and final post-repair admission, together with the dominant rejection reasons and the amortized indexing and synthesis cost.

Context quality. We sample Work Contexts for blind human review along four dimensions: professional realism, internal coherence, contextual sufficiency, and task openness. Reviewers inspect whether each core file has a defensible role, required dependencies and histories are present, distractors remain plausible, and the context can support multiple tasks without encoding a predetermined solution. We report per-dimension ratings, pairwise preferences against simpler construction baselines, and inter-annotator agreement.

C.3 Task Executability and Verifier Reliability

Task quality and executability. We evaluate the five checks used by Task Reviewer Agent: uniqueness, readability, contextual grounding, execution consistency, and task openness. For each check, we report the first-pass acceptance rate, average number of rewrites, final acceptance rate, and unresolved-candidate discard rate. Executability is measured by the fraction of tasks for which Demonstrator Agent reaches normal termination using only the supplied context and available tools, without Harness or tool-availability failures. A ready task must additionally make the Starter State fail and the Reference State pass under its compiled Verifier. We repeat executions from the same snapshot to test Harness reset, canonical state-diff, and evidence-capture reproducibility, and report invalid episodes separately from task failures.

Candidate-stage FCR and synthesis pass rate. To isolate FCR's effect on initial construction, we preserve a frozen stream of N_0 first-attempt Task Candidates and keep candidate identity fixed across methods. We report the shared no-repair first-attempt rate separately. Every candidate-visible Phase II or III failure is independently classified as policy-only, infrastructure-level, repairable under the frozen Contract anchor, or non-repairable. On the repairable subset, candidate-stage FCR is compared with discard/no repair, Local-only, Greedy-sequential, and full regeneration under matched proposal-token, rollout, terminal-Verifier, and wall-clock caps. A regeneration remains an attempt for the same initial candidate and never enlarges the denominator. Each method receives only candidate-visible feedback and may query the concealed initial-admission shard once. The primary endpoint is $\mathrm{SynthPass}_B$ from Equation 27; we additionally report recovery conditional on an initially failed repairable candidate, unresolved discard, intent and solver-effort drift, invalid-attempt rate, and cost. Thus any claimed synthesis gain is a difference in final admission under a fixed post-attempt repair budget, not a relabeling of the first-attempt pass rate. The complete candidate funnel and matched-budget comparison are reported in Table 29.

Counterfactual Verifier calibration. On the sealed audit cases, we report $\min_k \mathrm{CritAcc}_{i,k}^{\mathrm{audit}}$ together with family-level $\mathrm{MutScore}_{i,c}^{\mathrm{audit}}$, its macro-average and worst-family value, and $\mathrm{AltPass}_i^{\mathrm{audit}}$ for semantically valid solutions produced by distinct traces or semantics-preserving transformations. The same audit mutants and blinded human adjudication estimate false-pass and false-reject rates, with breakdowns by artifact type, evidence surface, and difficulty. We also report criterion evidence coverage, negative-mutant coverage, protected-state detection, cross-file input-ablation success, and the admission yield under Equation 17. We compare the complete Verifier with five precisely scoped variants: Judge-only routes all non-health decisions through the same single batched Judge; deterministic-only is compared on the paired subset of tasks whose criteria are all executable; unscoped-evidence gives that Judge the full evidence bundle; negative-only removes positive controls; and no-counterfactual-calibration removes both positive and negative controls.

Judge reliability and QA boundaries. On a stratified audit set, human reviewers independently label residual semantic criterion completion from the same frozen, criterion-tagged projected evidence shown to the Judge. Instruction and tool quality are audited separately as diagnostic process measures. We report semantic-criterion agreement, score correlation, threshold-decision agreement, abstention resolution, and repeated-Judge variance, with breakdowns by evidence surface and calibrated difficulty. Evidence-ablation and rubric-paraphrase tests measure sensitivity to missing tool results and superficial rubric wording. Structural and lineage QA yield,

`lineage_training_ready`, and manifest completeness are reported in a separate table: they characterize training-record integrity and sampling coverage, not task completion.

Audit cohort, labels, and uncertainty. The reporting unit is a labeled criterion–evidence case, nested within its task package and Work Context. The audit manifest must supply distinct task and Work Context counts, cases per criterion, class balance, sampling probabilities, and distributions over role, Harness, artifact type, and pre-calibrated difficulty. These counts are not supplied by the headline 95% criterion accuracy and 97% alternative acceptance figures. Those figures are reported point summaries, not confidence lower bounds or guarantees for every safety-critical or semantic condition.

Two annotators independently label completion from the frozen instruction, criterion, and projected evidence, without seeing Verifier decisions or repair identities. A third annotator adjudicates disagreements. Record annotator expertise, pre-adjudication agreement, adjudication counts, and unresolved labels. Calibration, initial admission, sealed reporting audit, and promotion banks remain disjoint at the recipe–seed level; reporting labels must not tune the Verifier, dependency rule, or policy.

Ordinary criteria are separated into deterministic functional, deterministic cross-artifact, and semantic groups. Integrity, protected-state, and safety gates are reported separately rather than hidden in a pooled accuracy. For evaluable cases with adjudicated truth $y \in \{0, 1\}$ and decision $\widehat{y}$, define TP/FN on $y = 1$ and TN/FP on $y = 0$. Then

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \qquad \mathrm{FRR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}, \qquad \mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

Unavailable evaluations and unresolved human labels are counted separately, with reasons; zero denominators yield unavailable rates. Report Work-Context-clustered bootstrap 95% intervals, task-macro and case-micro summaries, and worst-stratum values with their denominators. Rare strata with too few independent Work Contexts are descriptive only. Adversarial oversampling is reported as a separate challenge cohort or reweighted using the recorded sampling probabilities. Zero observed gate errors alone do not establish a zero population error rate.

Table 27: Required Verifier audit breakdown.

Criterion group	TP	FN	FP	TN	Invalid/unresolved	FAR [95% CI]	FRR [95% CI]
Deterministic functional	495	5	5	495	4	1.0 [0.4, 2.8]	1.0 [0.4, 2.8]
Deterministic cross-artifact	487	13	10	490	8	2.0 [1.0, 4.1]	2.6 [1.4, 4.9]
Semantic	485	15	35	465	15	7.0 [4.7, 10.3]	3.0 [1.6, 5.4]
Integrity gate	498	2	1	499	2	0.2 [0.0, 1.5]	0.4 [0.1, 1.8]
Protected-state gate	497	3	2	498	3	0.4 [0.1, 1.8]	0.6 [0.2, 2.2]
Safety gate	495	5	5	495	5	1.0 [0.4, 2.8]	1.0 [0.4, 2.8]

Valid alternatives and error examples. Positive controls come from independent solution traces or transformations that preserve all declared requirements while changing nonessential artifact layout, row order, filenames where unconstrained, or tool sequence. Record the generator/checkpoint or transformation recipe and its seed, then obtain independent human labels before applying the Verifier. Being produced by a particular model, or passing the Verifier under test, is not a validity label. Each observed false acceptance or rejection must link to its instruction, adjudicated criterion, evidence snapshot, and Verifier version. Prespecified failure probes include a correct aggregate with incorrect row lineage, a stale source version, and a protected-file change; valid-control probes include equivalent row orderings and alternative unconstrained output layouts. These describe audit fixtures.

Residual Judge manifest. Invocation rate is the fraction of evaluation-valid trajectories requiring semantic judgment, while semantic weight is the fraction of total criterion weight assigned to those judgments. Both are distinct from the upper bound of one batched Judge call. On the same frozen evidence, repeat each semantic judgment five times and compare with blinded human labels, reporting agreement, threshold flips, invalid outputs, and variation by criterion type. Deterministic and semantic criteria receive separate error estimates.

C.4 Long-Horizon Cowork Agent Performance

Table 1 in the main text gives the primary cross-benchmark comparison; Appendix C.1 specifies the published reference protocols and the controlled evaluation setup. Separately, we evaluate current agents on the WorkRover test set and its P1, P2, and P3 construction profiles. We stratify these

tasks by the empirical difficulty bands assigned before evaluation by the frozen reference panel; none of the evaluated agents contributes to those labels. Alongside WORKROVER hard-gated pass rate, we report S_B, diagnostic S_A, hard-gate violations, invalid-episode rate, criterion completion, interaction length, and execution cost. We then stratify performance by the number of files modified, workflow horizon, cross-root dependencies, noise ratio, occupational category, and artifact composition. This analysis tests whether failures are driven by workspace scale alone or by the relational and procedural complexity that scale induces. Repeated-rollout consistency further measures whether an agent can reproduce a Verifier-accepted outcome rather than benefit from a single favorable trajectory.

For task i and repeated rollout r, we summarize weighted criterion completion as

$$C_{i,r} = \frac{\sum_{k=1}^{K_i} w_{i,k} y_{i,r,k}}{\sum_{k=1}^{K_i} w_{i,k}}, \tag{36}$$

where $y_{i,r,k}$ is the thresholded deterministic or semantic decision for criterion k. The semantic subset is calibrated separately against human judgments.

C.5 Ablation Study of Verifier-to-Turn Credit Assignment

Primary matched-rollout-budget comparison. The main-text experiment compares SFT + VTCA-GRPO with SFT + Episode-only GRPO on all five external benchmarks in Figure 6. Both use the same SFT initialization and share the task and seed schedule, rollout-group size, attempted-rollout budget (including invalid, resampled, and skipped attempts), generated-token and optimizer-update budgets, interaction limits, and terminal Verifier/Judge-call budget. Training and tuning budgets are fixed on development tasks. VTCA retains one terminal Verifier pass and at most one terminal Judge call per valid trajectory. Intermediate evaluation uses only frozen deterministic progress checks. We separately log checker calls, CPU and wall-clock time, state-capture storage and time, and training compute; a companion run matches the total compute budget, allowing fewer rollouts when checks consume that budget.

Paired inference and mechanism endpoints. Core conditions A–H in Table 22 use the same five training-seed IDs and fixed SFT checkpoint. Their mechanism endpoints are the sealed WORKROVER test set, JobBench, and OfficeQA Pro. B versus F is the primary contrast on all five external suites in Figure 6, including AutomationBench. Training replicas have independent optimization and rollout randomness; repeated evaluations of one checkpoint are not training seeds.

For each contrast, pair task outputs and seed IDs across methods. Bootstrap training seeds and, within the paired evaluation, task or Work Context clusters using common resampling indices across methods; retain all repeats of a task. Use 10,000 resamples and report the mean difference with its 95% interval. Tests must respect the same clustering and seed pairing; individual turns and tokens are never independent replicates. Apply Holm–Bonferroni correction separately to B–F across all five external suites and to E–C/E–D across the three mechanism endpoints. Other contrasts and sensitivity sweeps are secondary. Report raw and adjusted p-values when the run-level records support the test; an interval crossing zero does not establish a reliable benefit.

GDPval uncertainty requires the original paired judgments under the same opponent pool, anchor, Judge, and presentation-order protocol. Resample tasks and training replicas, retain associated comparisons, and refit the rating model for each resample with the same anchor. Published Elo values from different snapshots cannot supply these intervals.

Benchmark endpoints and supplementary diagnostics. The primary endpoints are the five benchmark scores in Figure 6, evaluated at a fixed training budget. Relative to SFT + Episode-only GRPO, SFT + VTCA-GRPO gains 3.4 percentage points on JobBench, 43 Elo on GDPval v2, 5.7 percentage points on AutomationBench, 2.6 percentage points on WorkBuddy (Office), and 5.4 percentage points on OfficeQA Pro. These are the matched-comparison gains across task horizons and domains discussed in Section 4.3.1; individual VTCA-component effects are assessed by the supplementary mechanism contrasts. The supplementary internal audit reports official hard-gated pass rate and paired differences on in-distribution tasks, unseen Work Contexts, held-out roles, and held-out artifact compositions, with breakdowns by horizon, dependency depth, and calibrated difficulty. Secondary outcomes are S_B, S_A, criterion completion (Equation 36), $\Pr(G = 0)$ and $\Pr(H = 0)$ on valid episodes, invalid-attempt rate, tool errors, interaction turns, policy tokens, and execution cost. Learning-curve area under the pass-rate curve, indexed by attempted rollouts and terminal Verifier calls, measures sample efficiency. A companion curve against total compute measures cost effi-

ciency; across-seed dispersion measures training stability. Table 28 specifies these supplementary internal diagnostics. Figure 15 illustrates the corresponding training diagnostics. We treat entropy as a training-health diagnostic rather than an objective to minimize: prior RLVR studies associate entropy collapse with premature reward saturation, while also finding that excessive entropy can indicate an unstable, overly random policy (Cui et al., 2025; Li et al., 2025). Collection-yield and policy-drift diagnostics are deferred to Appendix B.8.

Table 28: Supplementary internal diagnostics for RQ4 on the 35B-A3B model. The primary benchmark scores are in Figure 6. The matched contrast is SFT + VTCA-GRPO versus SFT + Episode-only GRPO; other rows separate SFT and episode-level RL effects.

(a) Official hard-gated pass rate (%)

Method	**Overall**	**ID**	**Unseen WC**	**Held-out role**	**Held-out artifact**
Backbone	22.0	28.0	22.0	19.0	19.0
SFT	42.0	49.0	43.0	38.0	38.0
Backbone + Episode-only GRPO	32.0	39.0	33.0	28.0	28.0
SFT + Episode-only GRPO	54.0	62.0	55.0	49.0	50.0
SFT + VTCA-GRPO	59.0	67.0	60.0	55.0	54.0
Δ VTCA–Episode (SFT)	5.0	5.0	5.0	6.0	4.0

(b) Scores, gates, and evaluation health

Method	$\mathbf{S_B}\uparrow$	$\mathbf{S_A}\uparrow$	$\mathbf{C}\uparrow$	$\mathbf{Pr(G=0)}\downarrow$	$\mathbf{Invalid_{att}}\downarrow$
Backbone	0.410	0.700	0.380	0.460	0.050
SFT	0.610	0.820	0.580	0.300	0.040
Backbone + Episode-only GRPO	0.510	0.770	0.470	0.360	0.045
SFT + Episode-only GRPO	0.700	0.870	0.670	0.200	0.035
SFT + VTCA-GRPO	0.750	0.910	0.720	0.160	0.030
Δ VTCA–Episode (SFT)	0.050	0.040	0.050	-0.040	-0.005

Pass is the status in Equation 11, averaged over valid episodes; invalid attempts are reported separately over all attempts. S_B, S_A, C (Equation 36), and $\Pr(G=0)$ use the valid denominator. $G=1$ is necessary but not sufficient for pass. Entries report point estimates only; confidence intervals are not shown. Δ is the paired SFT-initialized difference.

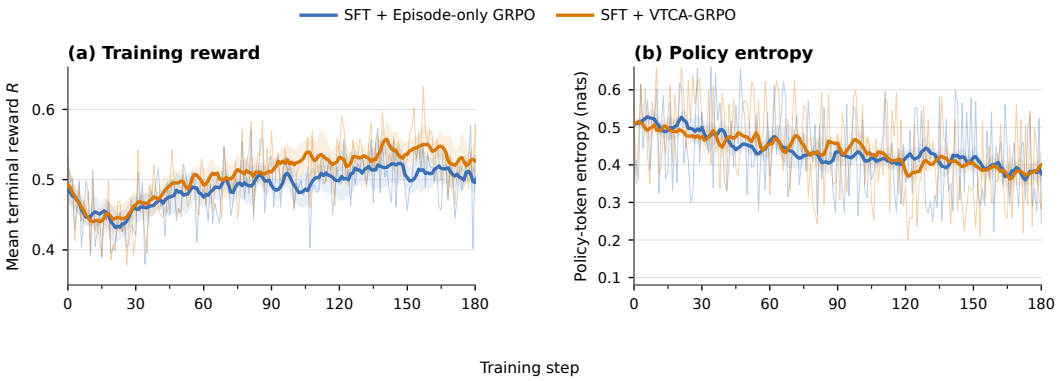

Figure 15: VTCA training dynamics: the shared terminal hard-gated reward R and policy-token entropy. In (a)–(b), faint traces are unsmoothed per-update minibatch estimates and thick lines are centered 3-update moving averages of the five-seed means. Bands are the corresponding 95% hierarchical-bootstrap intervals. The x-axis is training step (one optimizer update per step), showing the first 180 steps. Entropy is averaged over policy-generated tokens within turn, rollout, and seed; the auxiliary advantage-shaping term is not plotted as a reward.

Supplementary mechanism checks. The main text reports aggregate scores on five benchmarks. The specified component ablations separate local progress, dependency propagation, trace costs, turn balancing, and integrity gating. Centering is an auxiliary control, not the default. A sealed audit checks progress signs, prerequisite coverage, and the optimizer-facing shaping signal, including incomplete but constraint-valid trajectories. Realized budget and signal-health diagnostics and independent Verifier robustness checks are specified in Appendix B.8 (Tables 22–25).

Table 29: Candidate-stage synthesis recovery on the same frozen stream of N_0 initial candidates and under matched post-attempt repair budget B. First-attempt pass and routed share are common cohort quantities and are repeated to make the accounting explicit. Conditional recovery is computed only over the N_R independently confirmed repairable failures; final $\mathrm{SynthPass}_B$ retains N_0 as its denominator. The concealed initial-admission result is used once as a binary gate and never as repair feedback. All rate columns are percentages.

Method	**First attempt** $\mathrm{SynthPass}_0$	**Routed share** N_R/N_0	**Cond. recovery** N_{rec}/N_R	**Final** $\mathrm{SynthPass}_B$	**Intent drift**↓	**Rel. cost**↓
Discard / no repair	49.8	40.0	0.0	49.8	0.0	0.45
Local-only	49.8	40.0	32.5	62.8	2.0	0.70
Greedy-sequential	49.8	40.0	42.5	66.8	3.5	0.90
Full-regeneration	49.8	40.0	47.5	68.8	10.0	1.20
FCR	49.8	40.0	62.5	74.8	1.3	1.00

C.6 Analysis of Recursive Environment Evolution

Candidate-stage synthesis recovery. Table 29 separates the shared first-attempt yield from recovery after a candidate-visible failure. Let N_R be the frozen subset of the N_0 initial candidates that fail Phase II or III, are confirmed independently as Contract-preserving and repairable, and are routed to the compared repair method. The first-attempt rate and N_R/N_0 are therefore cohort quantities shared across methods; conditional recovery uses N_R as its denominator, whereas both the first-attempt and final synthesis pass rates use N_0. For FCR, this gives the auditable funnel $\mathrm{SynthPass}_0 \to N_R/N_0 \to N_{\mathrm{rec}}/N_R \to \mathrm{SynthPass}_B$. A repaired candidate must traverse every remaining construction gate and its untouched one-time initial-admission shard before it contributes to the final column.

Maintenance-stage failure cohorts. Separately from the candidate-stage synthesis experiment, we evaluate post-admission environment evolution on two disjoint, Work-Context-stratified cohorts. The natural cohort consists of consecutively screened task packages for which a fresh trigger rollout exposes a construction mismatch confirmed independently of the repair methods. Packages that remain Contract-consistent and reproducibly solvable by the Demonstrator are retained as policy-failure controls and cannot authorize a construction update. The controlled cohort starts from admission-clean, reproducibly healthy parents and injects preregistered, Contract-preserving single-, pairwise-, and higher-order defects into C, T, H, or V. The injection labels are concealed from diagnosis and proposal generation, giving ground truth for component attribution and over-repair. The complete 10K-task pool supplies the sampling frame and screening statistics; factorial replay is run only for confirmed triggers, not for every task. Appendix B.5 specifies the allocation, defect library, and human adjudication protocol.

Matched-budget repair comparison. The primary comparison includes the unchanged parent, *Local-only* repair of the highest-confidence ledger component, *Greedy-sequential* repair with replay after each accepted edit, *Full-regeneration* of the package, and FCR. In a candidate-controlled track, Local-only, Greedy-sequential, and FCR receive the same frozen direct proposals, isolating the effect of joint compatibility selection. A separate end-to-end track lets each method generate its own revisions. Full-regeneration preserves the same Evaluation Contract and receives the same total proposal-token and rollout budget; every method uses the same trigger evidence, parent lineage, rollout policy, and matched seed schedule. We additionally report Apply-all in Table 30 and an analysis-only Oracle-cell upper bound in Appendix B.5.

Primary maintenance effectiveness and safety outcome. For method A, let $\widehat{\mathcal{L}}_{i,A}$ be its proposed closed output for trigger i. Our preregistered primary metric is the *Safe Resolution Rate*

$$\mathrm{SRR}(A) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \mathbf{1}\left[\begin{array}{c} \mathrm{Resolve}_i(\widehat{\mathcal{L}}_{i,A}) = 1 \ \wedge\ \mathrm{Confirm}_i(\widehat{\mathcal{L}}_{i,A}) = 1 \\ \wedge\ \mathrm{AuditOK}_i(\widehat{\mathcal{L}}_{i,A}) = 1 \ \wedge\ \mathrm{TaskReg}_i(\widehat{\mathcal{L}}_{i,A}) = 0 \end{array}\right], \tag{37}$$

where $\mathcal{I}$ is a frozen set of confirmed construction-mismatch triggers. The following definitions specify the six columns in Table 30. Each trigger contributes one reported output per method, fixed before the sealed audit is opened; an unchanged parent is the output when no update is returned. Intermediate proposals and replay seeds do not add observations. Diagnostics concern these outputs, including outputs that fail final audit, and are not restricted to safely repaired or promoted packages.

Cohorts and Natural / Controlled SRR. Let $\mathcal{I}_{\mathrm{nat}}$ and $\mathcal{I}_{\mathrm{ctrl}}$ be the disjoint natural and controlled trigger sets, with achieved sizes N_{nat} and N_{ctrl}. Healthy and policy-failure controls, candidate-stage failures, and unscreened members of the 10K-task pool are excluded. Natural SRR and

Controlled SRR apply Equation 37 to these sets separately and multiply the resulting fractions by 100. $\mathrm{Resolve}_i = 1$ requires the predeclared typed defect predicates and independent runtime-health checks to pass; it is not defined by policy success or the candidate Verifier's score. $\mathrm{Confirm}_i = 1$ requires the selected output to pass the frozen confirmation rules on fresh seeds disjoint from proposal generation and selection replay. The confirmation seed count and acceptance thresholds belong to the frozen run configuration. Triggers remain in the denominator when unresolved, when their budget expires without a verified output, or when required checks are unavailable; these outcomes cannot establish a safe resolution.

For the single Audit OK, Task reg., and Rel. cost columns, the reporting population is the fixed union $\mathcal{I}_{\mathrm{all}} = \mathcal{I}_{\mathrm{nat}} \cup \mathcal{I}_{\mathrm{ctrl}}$, of size $N = N_{\mathrm{nat}} + N_{\mathrm{ctrl}}$. Each trigger has equal weight: pooled rates weight the two cohorts by their achieved sizes, rather than averaging their percentages equally. These diagnostics describe this evaluation mixture, not the prevalence of defects in natural deployment. Intent drift uses the adjudicated review sample drawn from this union, as specified below. Both comparison tracks retain their own outputs and resource ledgers; their results are not pooled.

Audit OK: sealed-audit acceptance. Write $a_{i,A} = \mathrm{AuditOK}_i(\widehat{\mathcal{L}}_{i,A})$. This binary indicator is one only if the output meets every applicable frozen audit condition: minimum criterion agreement, worst-family detection of Contract-entailing negative cases, acceptance of valid alternative solutions, and absence of unresolved evaluation-invalid cases. The corresponding thresholds $\alpha_{\mathrm{crit}}, \alpha_-, \alpha_+$ and case definitions are those of Appendix B.3, evaluated on the sealed audit shard. The reported percentage is

$$\mathrm{AuditOK}(A) = \frac{100}{N} \sum_{i \in \mathcal{I}_{\mathrm{all}}} a_{i,A}.$$

Audit OK counts packages satisfying all thresholds; it is not the average agreement across individual audit cases and does not by itself establish trigger resolution. Sealed audit outcomes are used only for final reporting, never for proposal generation, selection, or promotion. Missing required audit evidence cannot be counted as an audit pass.

Intent drift: change in task meaning. For an audited parent–output pair, $d_{i,A} = 1$ if adjudication finds a change to the professional objective, required outcome, intended state transition, or applicable semantic constraints. Correcting a binding or restoring declared behavior while preserving these requirements is not drift. A change in solver success or effort alone is a separate diagnostic. Let $\mathcal{R}_A$ contain the pairs with completed blinded adjudication under Appendix B.5. Then

$$\mathrm{IntentDrift}(A) = \frac{100}{|\mathcal{R}_A|} \sum_{i \in \mathcal{R}_A} d_{i,A}.$$

The numerator counts non-equivalent pairs after disagreement resolution; the denominator includes all adjudicated pairs, including unchanged outputs and unsuccessful repairs, rather than only successful repairs. The percentage describes the reviewed sample; a full-cohort estimate would additionally require the sampling weights. Report reviewed and unresolved counts by method and cohort. Unreviewed or inconclusive pairs are not assigned a zero-drift label, and an empty adjudicated sample yields an unavailable result.

Task reg.: regression in other retained packages. Let $\mathcal{F}_{i,A}$ contain the other retained packages affected by the output's direct or dependency-induced changes. A shared Harness change expands this set to all dependent retained packages. Write $f_{i,A} = \mathrm{TaskReg}_i(\widehat{\mathcal{L}}_{i,A})$; it is one if at least one member fails a frozen health, structural, Reference, or acceptance condition that its corresponding parent version satisfied under matched replay. It is zero when all required checks show no new failure. Pre-existing failures do not constitute new regressions. With complete affected-package checks, the rate is

$$\mathrm{TaskReg}(A) = \frac{100}{N} \sum_{i \in \mathcal{I}_{\mathrm{all}}} f_{i,A}.$$

A trigger counts once regardless of how many related packages regress; the denominator is not the number of related packages. An unchanged output or a verified empty affected set has no induced regression. If no regression is observed but required checks remain missing, the trigger's regression status is unknown. Report the count of such triggers and, until resolved, lower and upper bounds obtained by treating their indicators as zero and one, respectively. Such unknowns cannot satisfy the no-regression condition in SRR.

Rel. cost: normalized model-token usage. For trigger i, let $\mathcal{C}_{i,A}$ contain the model calls charged to method A, covering proposal generation, model-assisted rematerialization, selection replay, and fresh-seed confirmation. Include all model-based Verifier/Judge calls used for calibration, offline evidence scoring, and promotion, as well as solver-panel and affected-workspace validation calls used

to decide repairs. Count complete input and output usage, including tool results when resubmitted as input, reasoning tokens included in output usage, failed attempts, rejected proposals, and capped retries. Cached input tokens count once within each call's input total, without a monetary cache discount; usage subcategories are not added again to totals. Shared frozen proposals are charged once per trigger to each compared method, not once per replay cell. Define

$$t_{i,A} = \sum_{c \in \mathcal{C}_{i,A}} \left(n_c^{\text{in}} + n_c^{\text{out}}\right), \quad \bar{t}_A = \frac{1}{N} \sum_{i \in \mathcal{I}_{\text{all}}} t_{i,A},$$

$$\text{RelCost}(A) = \frac{\bar{t}_A}{\bar{t}_{\text{FCR}}}.$$

The FCR reference uses the same cohort and comparison track, so its point estimate is 1.00; 1.34 denotes 34% more model tokens. All attempted work is charged even when no repair is returned. Deterministic rematerialization, CPU/GPU time, tool execution, and human effort are recorded separately and are not converted into tokens. Final independent audit calls are also kept in a separate ledger. Token usage measures model-call volume, not monetary cost or hardware compute. The ledger records model/tokenizer identifiers; missing usage is not zero. Equal budget caps need not imply equal realized usage.

Intervals and reporting records. All rate columns are percentages; Rel. cost is a dimensionless ratio. Parentheses denote 95% workspace-clustered bootstrap intervals, except for the Controlled Parent boundary interval specified below. Method comparisons are paired on trigger identifiers. Resampling retains the distinction between natural and controlled cohorts and their fixed weights in pooled estimates; replay seeds are nested observations, not independent samples. Intent-drift intervals use the adjudicated sample and its review strata. For normalized-cost intervals, bootstrap each method's mean token usage while holding the observed FCR reference mean fixed. This convention allows a nondegenerate interval around FCR's normalized mean of 1.00; it describes uncertainty in normalized usage, not uncertainty in a ratio with a resampled reference denominator. Paired cost contrasts use the same workspace resamples.

The Controlled Parent SRR entry in Table 30 is a boundary-case exception. With zero safe resolutions among 260 controlled triggers, ordinary nonparametric resampling of the fixed binary outcomes gives $[0, 0]$. For this entry, we compute the two-sided 95% Clopper–Pearson exact-binomial reference interval:

$$\left[0,\ 100(1 - 0.025^{1/260})\right]\% \approx [0.0,\ 1.4]\%.$$

This reference interval assumes independent Bernoulli triggers; it is not workspace-cluster-adjusted and is not used for paired method inference.

The reporting record must identify the achieved cohort sizes, each metric's numerator and denominator, review coverage and unresolved labels, frozen audit and confirmation configurations, and per-trigger token totals.

Selection under shared proposals. The candidate-controlled panel of Table 30 holds direct repair proposals fixed to isolate compatibility selection. Relative to Apply-all, FCR improves natural and controlled SRR from 64.0% to 73.0% and from 70.8% to 81.2%, respectively, and reduces cross-task regression from 7.0% to 0.7%, while using more model tokens (relative cost: 1.00 versus 0.59).

Joint revision and stability. Among safely resolved triggers, pairwise joint-necessity rates range from 0.48 to 0.74 for the frozen proposals (Figure 7(a)): the selected repair requires multiple direct revisions, and no strict subset satisfies the same repair criteria. Panel (b) shows natural-trigger safe-resolution rates. Across three construction generations (panel c), SRR stays within 70.3–72.4% and sealed-audit acceptance within 93.8–94.6%.

Necessity of joint revision. On controlled defects, we report ledger precision, recall, and macro-F1, exact recovery of the injected direct-component support, and unnecessary-component edits. On both cohorts, the *joint-necessity rate* is the fraction of safely resolved triggers for which the selected multi-component cell is admissible but every strict subset of its direct revisions is either inadmissible or fails the frozen resolution and promotion predicates. We also report the availability of complete compatible subcubes and the preregistered main and pairwise interaction contrasts from Appendix B.4, including unavailable contrasts rather than dropping them. Results are broken down by active-factor count, defect source, artifact composition, difficulty, and whether a Harness revision in the closure affects multiple packages.

Safeguards, generations, and statistical unit. The full safety comparison reports information-boundary violations, sealed anchor accuracy, task-intent and solver-effort drift, false repair on

Table 30: Full matched-budget maintenance-stage results. Natural and controlled SRR are reported separately; parenthesized values give 95% workspace-clustered bootstrap intervals, except for the †-marked Controlled Parent exact-binomial reference interval. Audit OK and Task reg. use the combined trigger cohort; Intent drift uses adjudicated review pairs. Rel. cost is defined by model-token usage relative to FCR within each panel. Definitions and ledger reconciliation status: Appendix C.6. The candidate-controlled panel isolates compatibility selection using shared frozen proposals; the end-to-end panel compares complete repair systems.

Method	Natural SRR↑	Controlled SRR↑	Audit OK↑	Intent drift↓	Task reg.↓	Rel. cost↓
Candidate-controlled: identical frozen proposals						
Local-only	42.0% (36.5–47.6)	45.4% (39.4–51.5)	81.1% (77.7–84.2)	2.0% (1.0–3.6)	3.2% (1.9–5.0)	0.43 (0.40–0.47)
Greedy-sequential	58.0% (52.4–63.4)	63.1% (57.1–68.8)	89.3% (86.4–91.7)	3.4% (2.1–5.3)	2.7% (1.5–4.4)	0.78 (0.72–0.84)
Apply-all	64.0% (58.5–69.3)	70.8% (65.0–76.1)	79.5% (75.9–82.8)	9.5% (7.2–12.2)	7.0% (5.0–9.5)	0.59 (0.55–0.64)
FCR	73.0% (67.7–77.8)	81.2% (76.0–85.7)	95.0% (92.9–96.6)	0.9% (0.3–2.1)	0.7% (0.2–1.8)	1.00 (0.93–1.08)
End-to-end: independently generated repairs under matched budgets						
Parent / no repair	3.0% (1.4–5.5)	0.0% (0.0–1.4)†	29.8% (26.0–33.8)	0.0% (0.0–0.0)	0.0% (0.0–0.0)	0.13 (0.12–0.14)
Local-only	39.3% (33.9–44.9)	40.8% (34.9–46.9)	78.4% (74.8–81.8)	2.5% (1.4–4.3)	3.8% (2.4–5.8)	0.61 (0.56–0.66)
Greedy-sequential	55.0% (49.4–60.5)	57.7% (51.6–63.5)	86.6% (83.5–89.3)	4.1% (2.7–6.2)	3.6% (2.2–5.5)	0.92 (0.85–1.00)
Full-regeneration	60.7% (55.1–66.0)	67.3% (61.4–72.8)	74.1% (70.2–77.7)	14.3% (11.4–17.6)	8.2% (6.1–10.8)	1.34 (1.23–1.46)
FCR	70.3% (65.0–75.3)	77.7% (72.2–82.4)	93.8% (91.4–95.6)	1.3% (0.5–2.6)	1.1% (0.4–2.4)	1.00 (0.93–1.08)

healthy or policy-failure controls, and escaped cross-task regressions. Appendix B.5 isolates exact closure, matched seeds, sealed promotion shards, information-boundary review, lexicographic minimality, cross-task regression, and fresh-seed confirmation through analysis-only ablations. On a workspace-stratified subset, we further repeat maintenance-stage FCR for three preallocated construction generations and report defect incidence, SRR, sealed-audit acceptance, closure size, marginal cost, and the fraction of the originally admitted cohort that remains audit-clean after each generation (retained validity). The statistical unit is the trigger package, with workspace-clustered paired bootstrap intervals; replay cells and seeds are repeated measurements rather than independent samples. Held-out-policy audits test whether promoted packages generalize beyond the rollout policy that exposed the trigger. Figure 7(a–c) summarizes the joint-necessity, safety, and longitudinal results; Appendix B.5 gives the corresponding denominators and diagnostic breakdowns. Factor-count costs and upstream diagnostic coverage are separately specified in Tables 11 and 9.

C.7 Construction Ablations and Error Analysis

Construction ablations. Table 3 presents the key construction comparisons in the main text; Table 31 retains all variants and quality metrics. We remove job-skill grounding, file-skill grounding, relation-aware expansion, realistic background sampling, validation-guided rewriting, and recursive environment evolution in turn. We measure their effects on context quality, task yield, reference executability, judge reliability, and downstream agent success. The comparison with directory-based, random, and semantic-only assembly tests whether the complete dual-skill pipeline contributes beyond topical file similarity or physical co-location. The reported success-rate comparisons condition on each variant's retained workspaces. They do not establish a fixed-intent effect: construction variants may admit different tasks and change their difficulty.

Fixed initial intents and denominators. Freeze initial intent IDs, source-material versions, required outcomes, evaluation constraints, and construction seeds before running any variant. All methods receive the same intents and resource caps; regeneration remains an attempt for the original ID. For method A and initial intent j, record construction, reference execution, independent Verifier validation, admission, and frozen-agent success separately. Each unconditional rate uses the same N_{intent} initial intents. In particular,

$$\text{EndPass}(A) = \frac{1}{N_{\text{intent}}} \sum_{j=1}^{N_{\text{intent}}} \mathbf{1}[\text{Admitted}_{A,j}]\mathbf{1}[\text{AgentPass}_{A,j}].$$

Unconstructed, rejected, and budget-exhausted intents add no end-to-end success and remain in the denominator, with their statuses reported. Infrastructure-invalid and unobserved outcomes have

Table 31: Construction ablations and workspace-assembly baselines. Context quality is the mean blind-review score on a 1–5 scale; all other columns are percentages. Entries are means with workspace-clustered 95% interval half-widths. Success Rate evaluates the same frozen agent on each variant's retained workspaces.

Construction method	Context quality↑	Task yield↑	Reference exec.↑	Judge agree.↑	Success Rate↑
WORKROVER (full)	4.43 ±0.07	74.8 ±2.20	95.1 ±1.10	92.4 ±1.5	58.7 ±2.60
One-component removals					
− job-skill grounding	4.08 ±0.09	67.2 ±2.38	91.7 ±1.48	90.5 ±1.7	53.1 ±2.78
− file-skill grounding	3.86 ±0.10	60.9 ±2.47	84.2 ±2.06	87.1 ±2.0	47.4 ±2.92
− relation-aware expansion	3.52 ±0.11	52.6 ±2.53	77.9 ±2.52	84.3 ±2.3	41.8 ±3.11
− realistic background sampling	3.76 ±0.10	72.1 ±2.3	94.0 ±1.3	91.7 ±1.6	55.2 ±2.7
− validation-guided rewriting	3.94 ±0.10	49.8 ±2.53	80.6 ±2.47	82.7 ±2.4	44.5 ±3.22
− recursive environment evolution	4.15 ±0.08	63.7 ±2.6	87.8 ±1.9	88.4 ±1.9	50.2 ±2.8
Workspace-assembly baselines					
Directory-based	3.28 ±0.12	44.9 ±2.8	69.8 ±2.8	79.6 ±2.5	34.9 ±2.7
Random	2.14 ±0.13	17.6 ±2.1	38.5 ±2.9	65.9 ±2.8	18.8 ±2.2
Semantic-only	3.45 ±0.11	48.7 ±2.53	73.6 ±2.78	81.4 ±2.4	38.6 ±3.19

separate counts, rather than being silently dropped. Conditional reference executability, Verifier validity, and agent success must state their own denominators alongside these unconditional rates.

Difficulty and common retained subsets. Use an independent frozen solver panel to assign reference difficulty to the shared initial intents before evaluating construction variants; none of the evaluated agents supplies these labels. Report matched differences within the same initial-intent difficulty strata and Work-Context-clustered intervals. Also report each variant's realized distributions of turns, tool calls, distinct skills, artifact counts, and dependency depth, with medians and interquartile ranges and the fraction of missing measurements. Variant-induced difficulty changes are outcomes, not grounds for removing intents from the primary comparison.

For supplementary conditional comparisons, use the intersection of retained intent IDs across all compared methods and report its size and coverage of the original cohort. Keep original outcome requirements fixed and evaluate with independent checks. An empty intersection yields no conditional estimate. Difficulty matching on retained outputs remains a secondary, selection-dependent analysis; it does not replace the full-cohort estimate.

Separating filtering from rewriting. Run three arms under the same intent stream: no validation feedback and no rewriting; validation-based admission filtering only; and the same filtering plus validation-guided rewriting. The first arm is a sandboxed construction control and still undergoes independent final evaluation. All arms preserve the original outcome constraints and use the same frozen downstream agent; actual proposal, execution, and validation costs are charged. Table 32 reports feasibility, Verifier validity, admission, and downstream success separately. Comparing filtering-only with filtering-plus-rewriting tests repair beyond exclusion, while the full intent denominator exposes any loss of yield.

Table 32: Fixed-intent construction controls. Conditional rates and realized difficulty distributions accompany the result ledger.

Construction control	N_{intent}	Reference exec.	Verifier valid	Admission	EndPass
No filtering or rewriting	10000	80.0	70.0	49.8	22.2
Filtering only	10000	80.0	70.0	49.8	22.2
Filtering + rewriting	10000	92.0	87.0	74.8	43.9

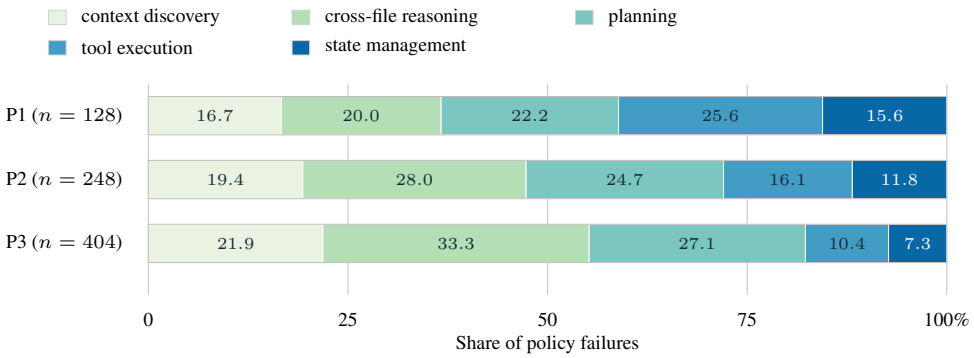

Figure 16: Policy-failure composition by construction profile for the frozen evaluation agent. The five plotted categories exclude Judge mismatch; shares are renormalized from rounded percentages. Percentages may not sum to 100 because of rounding. Representative trajectories help distinguish limitations of the evaluated policy from those of WORKROVER.