



WorkRover: Scaling Long-Horizon Cowork Agent Training Via Recursive Environment Evolution

Hang He*, Linting Wang*, Guanming Liu*, Bo Mao*, Jinxiu Liu*, Yiran Suo*,
Maosen Zhou, Zhiheng Xi, Guanting Dong, Zhou Shao✉, Rui Zheng, Jie Zhou,
Chengcheng Wan, Tao Gui✉, Qi Zhang, Xipeng Qiu, Xuanjing Huang

*Equal Contribution, ✉Corresponding Authors

Fudan University, East China Normal University, Renmin University of China,
Shanghai Qiji Zhifeng Co., Ltd., Shanghai Innovation Institute

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.

Project Page: <https://workrover.github.io/>

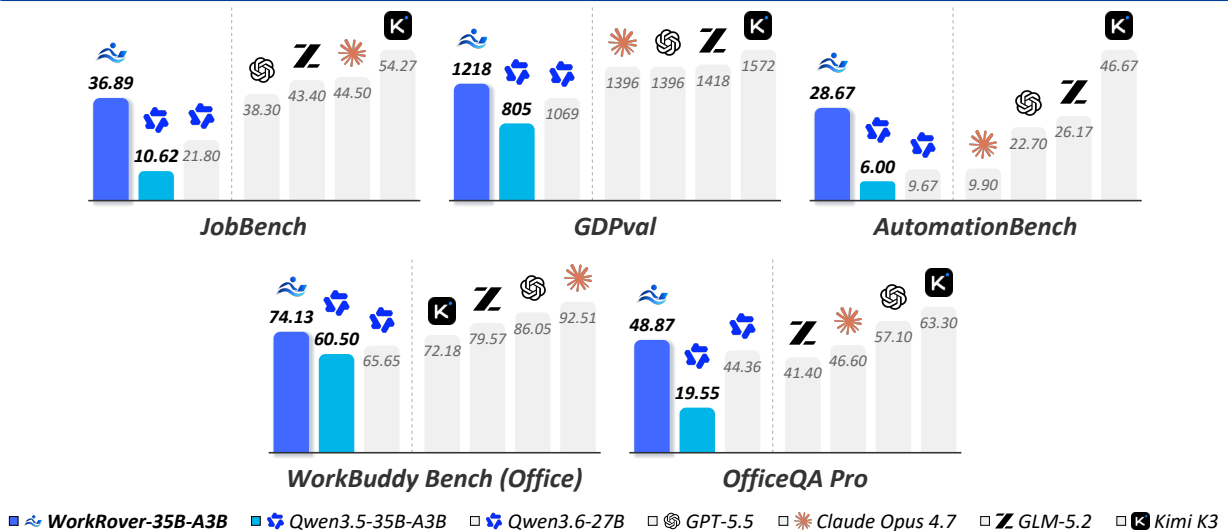


Figure 1 Cross-benchmark performance. Results on five cowork-agent benchmarks.

1 Introduction

Cowork agents powered by large language models (LLMs) perform professional tasks across files, tools, and applications [15]. These tasks unfold over long sequences of dependent actions, requiring agents to locate relevant materials [17, 31, 37], track workspace changes, and maintain consistency across related artifacts [3, 20, 32, 33]. Training these agents through reinforcement learning (RL) depends on environments that support diverse, executable workflows and provide reliable feedback on task outcomes [5, 10, 11, 26, 41, 45, 49]. Recent work has explored automated environment and task synthesis to expand the range of workspaces and tasks available for agent training [2, 18, 30].

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 [47]. 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 [6, 34].

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.

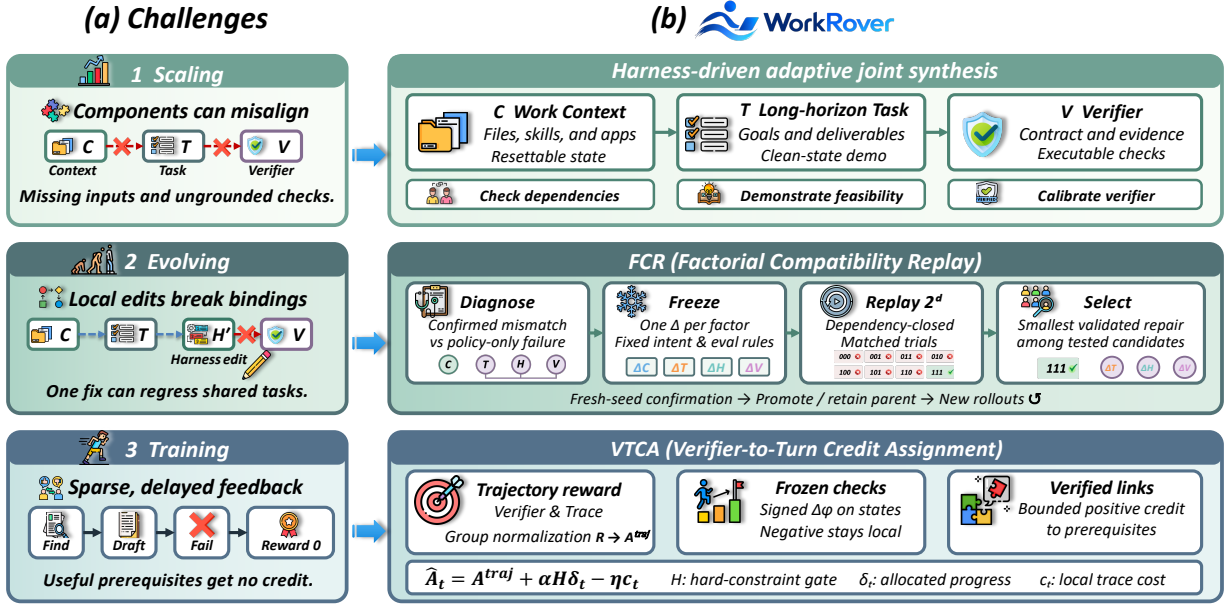


Figure 2 Cowork agents: challenges and approaches.

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 [2, 7, 30, 36, 40] and validated task synthesis [4, 12, 18, 22]. Shared specifications and state help align tasks and verifiers [13, 29], while adversarial audits test verifier reliability [47]. 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 repair [10, 23, 39]. Program repair also studies patch minimization [38]. 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 [8, 24], progress estimates, or evidence attribution [9, 34, 42–44, 46]. VTCA shapes GRPO advantages [27] with frozen checks of workspace progress and setbacks. It assigns a bounded share

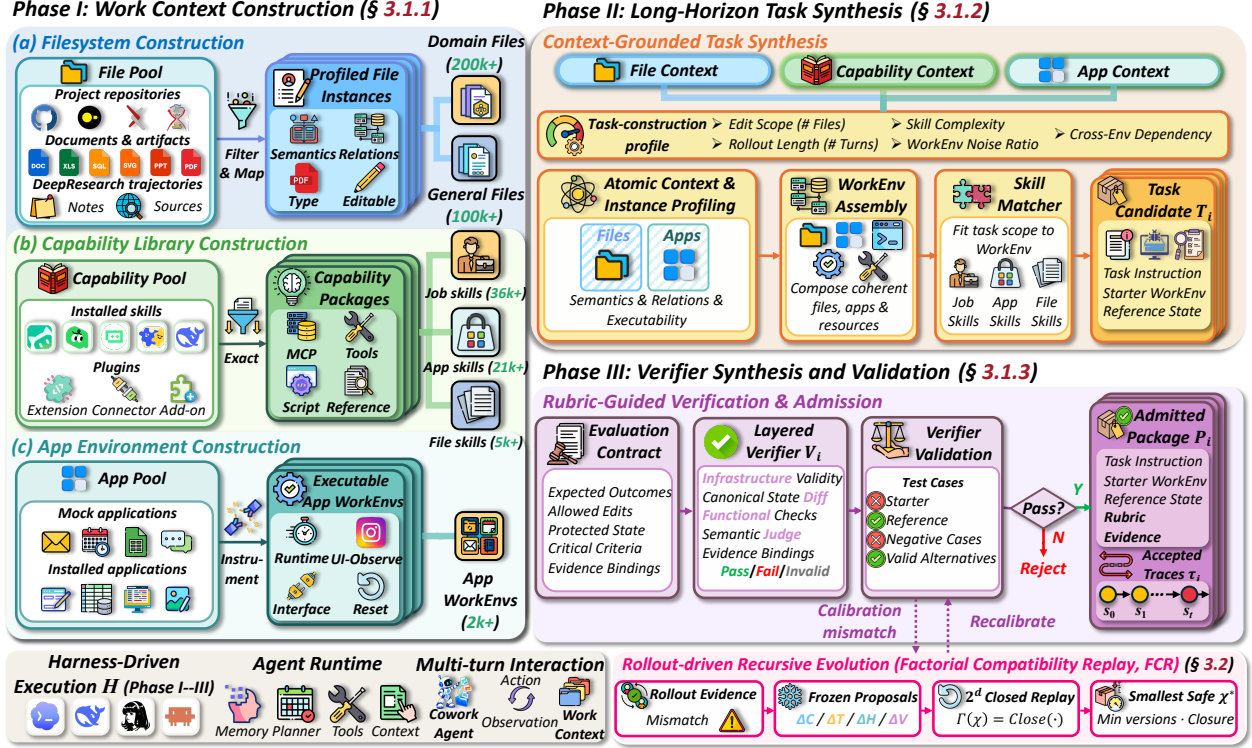


Figure 3 Environment construction and recursive evolution in WORKROVER. The Harness coordinates Work Context construction, Task synthesis, and Verifier synthesis and validation. FCR repairs compatibility mismatches under frozen safety gates; verified trajectories support policy learning.

of positive credit to prerequisite turns using recorded tool-output and artifact-version dependencies, without intermediate Judge calls. Checker coverage [48] and dependency quality limit these signals. See Appendix A.

3 Method

In this section, we present **WorkRover** (Figure 3), a framework for training long-horizon cowork agents through recursive environment evolution. We first introduce 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

We organize the training environments into N task packages:

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

The Work Context C_i provides files, capabilities, and application environments; the Task T_i specifies goals, deliverables, and constraints; and the Verifier V_i checks task completion. The Harness H coordinates construction, tracks dependencies, and manages execution across the three phases in Figure 3.

3.1.1 Work Context Construction

Phase I constructs a reusable Work Context C_i comprising File Context C_i^{file} , Capability Context C_i^{cap} , and App Context C_i^{app} . We collect files from project repositories and DeepResearch trajectories, along with documents and artifacts. After filtering, we profile their semantics, cross-file relations, type, and editability to form general-purpose and domain-specific collections. We package installed skills and plugins with their MCP interfaces, tools, scripts, and references. These packages form job, file, and app skill libraries. We

instrument mock and installed applications to create executable App WorkEnvs with a runtime, UI observation, interaction interfaces, and reset support. Together, these resources supply the three Contexts used for WorkEnv assembly and task synthesis in Phase II. Figure 4 summarizes skill-library coverage. Appendix B.1 provides construction details and complexity analysis; Appendix B.2 reports resource coverage.

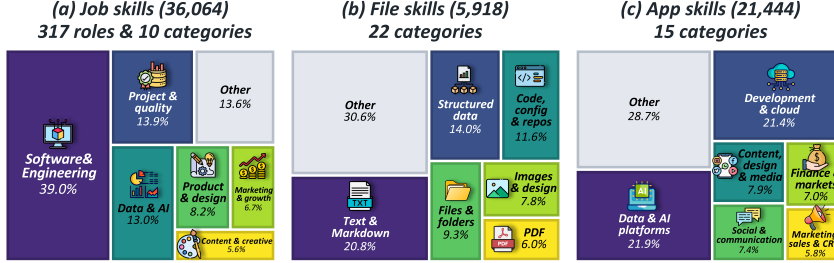


Figure 4 Skill-library coverage. Six largest categories; others aggregated. Tile sizes show shares of skills (a, c) or category memberships (b).

3.1.2 Long-Horizon Task Synthesis

Phase II synthesizes task T_i from the three Contexts in C_i (Figure 3). A construction profile sets the target edit scope (files), interaction horizon (turns), skill complexity (number and types of combined skills), distractor ratio, and cross-environment dependencies (Table 4; Appendix B.4). We profile selected file and app instances for task-relevant semantics, relations, and executability. The Harness H then assembles compatible files, apps, and capabilities into a WorkEnv. Starting from a job skill, the Skill Matcher fits the task scope to the WorkEnv’s available resources and combines job, file, and app skills into an internal workflow whose stages depend on earlier outputs or state changes. Its dependency graph guides preparation of the Starter WorkEnv x_{v_i} , including all required resources and marking editable and protected state. Task instructions specify goals, materials, deliverables, and constraints. An independent Demonstrator Agent executes the task from x_{v_i} ; success confirms feasibility and yields a Reference State x_i^* . The Task Instruction, Starter WorkEnv, and Reference State form the candidate, with state differences providing evidence of completion. The workflow and reference execution must not add requirements or exclude other valid solutions. We remove inputs to test dependency necessity; independently confirmed compatibility problems trigger FCR (Section 3.2). Candidates passing construction checks proceed to Phase III for Verifier synthesis and validation.

3.1.3 Verifier Synthesis and Validation

Phase III synthesizes and validates the Verifier V_i (Figure 3). We derive an Evaluation Contract from the task instructions and agent-visible workspace conditions, fixing its criteria before inspecting the Reference State. The Contract specifies expected outcomes, allowed edits, protected state, critical criteria, and admissible evidence. We use it to generate a rubric and verification rules, binding each criterion to evidence from files, application states, or state differences. *Infrastructure Validity* returns *invalid* when infrastructure faults prevent evaluation; *Canonical State Diff* checks state changes against the Contract; *Functional Checks* test artifact content and consistency; and *Semantic Judge* assesses criteria unresolved by executable checks. Valid evaluations pass only if the weighted outcome score meets its threshold and all mandatory checks pass; otherwise they fail. We calibrate V_i to reject the Starter State and negative cases that violate the Contract, while accepting the Reference State and independently obtained valid alternatives. Independently confirmed compatibility problems may trigger FCR, followed by recalibration under the same fixed Contract. We then freeze V_i for a separate, one-time admission test; candidates that fail are rejected without feedback for further repair. Passing candidates become admitted packages $\mathcal{P}_i$ containing the Task Instruction, Starter WorkEnv, Reference State, rubric, and evidence, with accepted execution traces τ_i retained for training. See Appendix C.1 for scoring and validation details.

3.2 Recursive Environment Evolution

The Work Context, Task, Harness, and Verifier must work together for a task to run and be evaluated correctly. *Factorial Compatibility Replay* (FCR) repairs mismatches revealed during construction or later

agent rollouts. Since changing one component can affect the others, FCR tests proposed changes both individually and jointly, then selects the smallest validated repair that preserves the task’s original intent.

Diagnosis and proposals. Independent diagnostics first distinguish environment problems from mistakes in the agent’s decisions. Agent mistakes alone do not trigger environment changes. For an environment problem, the Harness records the affected versions and dependencies, keeps task requirements and validation checks fixed, and freezes one proposed repair per eligible component: ΔC , ΔT , ΔH , and ΔV as applicable. The Evaluation Contract remains fixed once grounded in the task requirements. Edits visible to the agent must preserve information boundaries, including keeping hidden evaluation evidence hidden. These edits must also pass checks by a fixed panel of solvers for unintended task simplification.

Harness screening and replay. For $d \leq 4$ eligible components, each combination $\chi \in \{0, 1\}^d$ specifies which proposals to apply: 1 applies a proposal and 0 omits it. This gives 2^d combinations, at most 16. Before rebuilding packages or running rollouts, the Harness removes combinations with confirmed conflicts in dependencies, resource bindings, or requirements; uncertain cases proceed to validation. Each remaining combination is expanded into a dependency-closed update set $\Gamma(\chi) = \text{Close}(\chi)$. This set includes the selected repairs and all additional updates needed to keep dependent components compatible, even if those components were not directly selected for repair. The rebuilt packages are then replayed with the same agent policies, execution budgets, and random seeds. Comparing individual and joint repairs reveals whether some changes work only when applied together. If only the Verifier changes, all alternatives evaluate the same recorded evidence.

Repair selection. Independent checks must confirm that the original problem is resolved, required outcomes and information boundaries are preserved, and the repair introduces no new problems in affected task packages. These checks do not use agent success or scores from the proposed Verifier as evidence of resolution. Once a Verifier exists, the repair must also pass the fixed Contract checks and tests confirming that the Verifier rejects known failures and accepts valid alternative solutions. Among the tested repair combinations that pass these checks, FCR selects χ^* whose full update set $\Gamma(\chi^*)$ changes the fewest component versions, counting updates required by dependencies. Ties favor fewer rebuilt dependencies, then lower cost.

Confirmation and update. Fresh rollouts with new random seeds confirm the selected repair. A candidate that has not yet been admitted to the training collection repeats validation from the earliest affected construction step and must pass initial admission. For an already admitted package, all validated changes take effect together after confirmation. If repair selection or confirmation fails, FCR discards an unadmitted candidate or keeps the previous admitted version. Later rollouts can reveal new mismatches and initiate another repair cycle. See Appendices C.2 and C.3 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 5).

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 $A_{i,r}^{\text{traj}}$ for task i and rollout r .

Measuring progress. Frozen deterministic checks [34] 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.

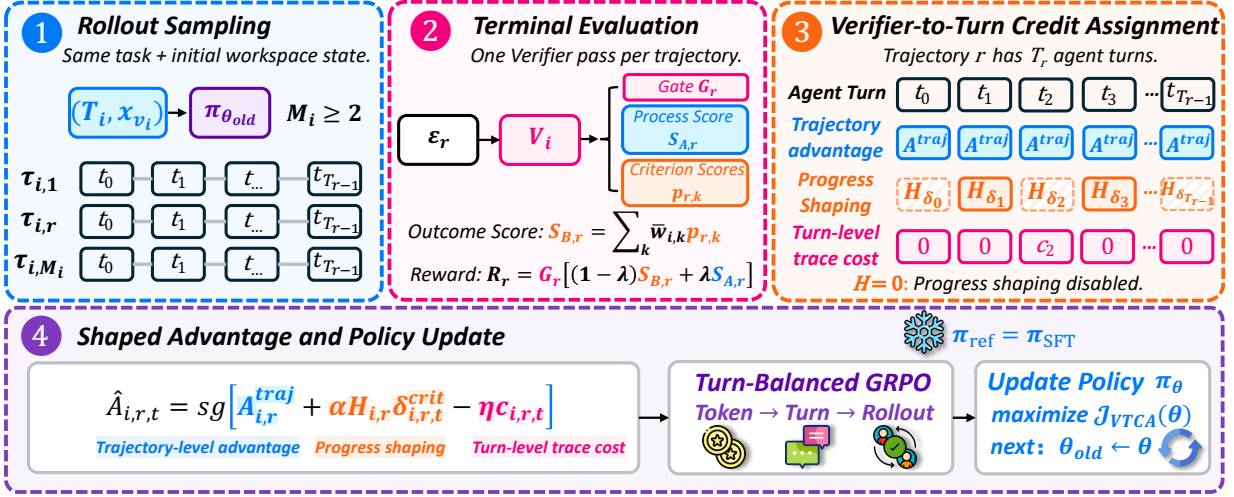


Figure 5 Overview of VTCA. Same-task rollouts yield terminal rewards and checked progress. Bounded prerequisite allocations and local trace costs shape turn-balanced GRPO advantages.

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 criterion-based credit $\delta_{i,r,t}^{\text{crit}}$ 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:

$$\hat{A}_{i,r,t} = \text{sg} \left[A_{i,r}^{\text{traj}} + \alpha H_{i,r} \delta_{i,r,t}^{\text{crit}} - \eta c_{i,r,t} \right], \quad \alpha, \eta \geq 0, \quad (1)$$

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 [27] 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 C.4).

4 Experiments

4.1 Experimental Setup

Model and training data. We post-train Qwen3.5-35B-A3B [25] to obtain WORKROVER-35B-A3B. Supervised fine-tuning (SFT) uses 30,115 trajectories generated by 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). Subset-selection details are in Appendix D.1.

Evaluation. We evaluate transfer to five external benchmarks: JobBench [16] for occupational tasks, GDP-val [1] for professional deliverables, AutomationBench [28] for cross-application workflows, WorkBuddy Bench (Office) [35] for office workflows, and OfficeQA Pro [21] for document-grounded question answering. We use

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: $\uparrow$ increases and $\downarrow$ decreases (Elo or percentage points). Higher scores are better. Details in Appendix D.1.

Model	JobBench (%) $\uparrow$	GDPval (Elo) $\uparrow$	Automation Bench (%) $\uparrow$	WorkBuddy Bench (Office) (%) $\uparrow$	OfficeQA Pro (%) $\uparrow$
<i>Proprietary Models</i>					
 GPT-5.5	<u>38.30</u> $\uparrow_{27.68}$	1395.85 $\uparrow_{591.10}$	<u>22.70</u> $\uparrow_{16.70}$	<u>86.05</u> $\uparrow_{25.55}$	57.10 $\uparrow_{37.55}$
 Claude Opus 4.7	44.50 $\uparrow_{33.88}$	<u>1395.78</u> $\uparrow_{591.03}$	9.90 $\uparrow_{3.90}$	92.51 $\uparrow_{32.01}$	<u>46.60</u> $\uparrow_{27.05}$
 Gemini 3.1 Pro Preview	15.90 $\uparrow_{5.28}$	903.94 $\uparrow_{99.19}$	37.54 $\uparrow_{31.54}$	—	39.10 $\uparrow_{19.55}$
<i>Open-Weight Models (>400B total parameters)</i>					
 Kimi K3	54.27 $\uparrow_{43.65}$	1572.42 $\uparrow_{767.67}$	46.67 $\uparrow_{40.67}$	72.18 $\uparrow_{11.68}$	<u>63.30</u> $\uparrow_{43.75}$
 DeepSeek-V4-Pro	<u>54.10</u> $\uparrow_{43.48}$	<u>1493.31</u> $\uparrow_{688.56}$	<u>31.80</u> $\uparrow_{25.80}$	<u>78.71</u> $\uparrow_{18.21}$	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 $\uparrow_{25.55}$
<i>Open-Weight Models (27–35B total parameters)</i>					
 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	<u>34.27</u> $\uparrow_{23.65}$	<u>1128.00</u> $\uparrow_{323.25}$	<u>27.60</u> $\uparrow_{21.60}$	<u>66.65</u> $\uparrow_{6.15}$	<u>48.10</u> $\uparrow_{28.55}$
 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 $\uparrow_{29.32}$

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. Our training and evaluation setup is described in Appendix D.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.

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 D.2 and C.4.

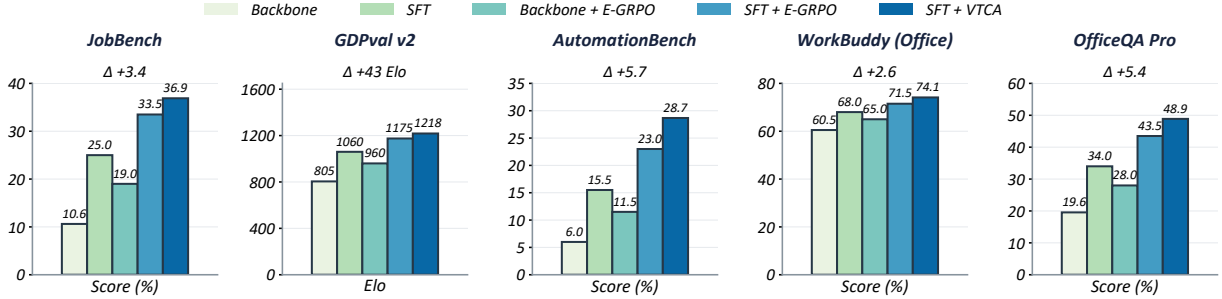


Figure 6 35B-A3B comparison under matched rollout budgets. Backbone denotes Qwen3.5-35B-A3B before our SFT or RL. Gains compare SFT + VTCA with SFT + E-GRPO. GDPval v2 uses Elo; other scores are percentages.

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).

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 pass rate at 93.8–94.6% (panel c), showing little change in repair quality across these rounds. FCR’s task success rate 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. Repair therefore recovers more tasks that pass validation from the same initial pool.

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,

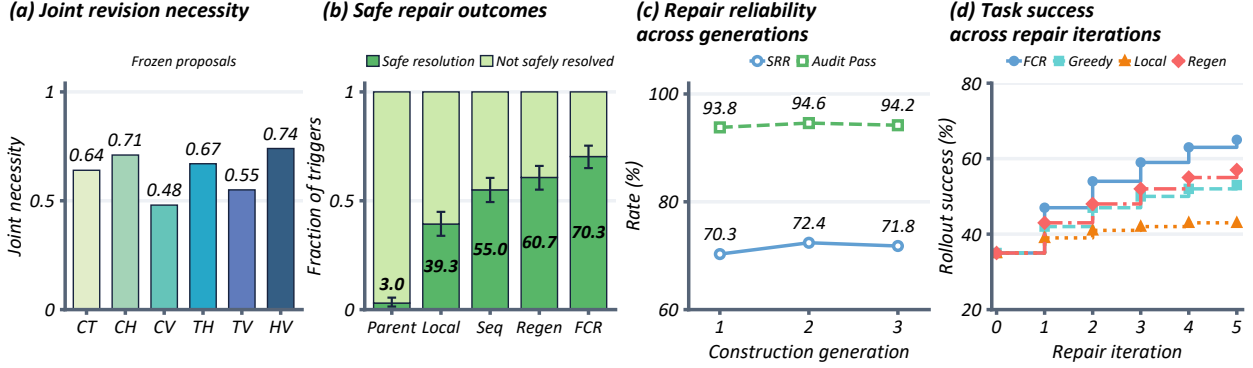


Figure 7 FCR mechanism, stability, and evolution. (a) Pairwise joint necessity (C: Context; T: Task; H: Harness; V: Verifier). (b) Natural-trigger outcomes. (c) Repair reliability across three generations, measured by SRR and sealed-audit pass rate. (d) Task success across five repair iterations.

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.

Method	Natural SRR $\uparrow$	Controlled SRR $\uparrow$	Task reg. $\downarrow$	Rel. cost $\downarrow$
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 $\pm$ workspace-clustered 95% interval half-width.

Construction	Task yield	Reference exec.	Success Rate
WORKROVER (full)	74.8 $\pm$ 2.20	95.1 $\pm$ 1.10	58.7 $\pm$ 2.60
<i>w/o Job-Skill</i>	67.2 $\pm$ 2.38	91.7 $\pm$ 1.48	53.1 $\pm$ 2.78
<i>w/o File-Skill</i>	60.9 $\pm$ 2.47	84.2 $\pm$ 2.06	47.4 $\pm$ 2.92
<i>w/o Relation Expansion</i>	52.6 $\pm$ 2.53	77.9 $\pm$ 2.52	41.8 $\pm$ 3.11
<i>w/o Validation Rewrite</i>	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. 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.

References

- [1] Artificial Analysis. GDPval v2 leaderboard, 2026. 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.
- [2] F. Bai, H. Song, S. Sun, D. Cheng, Y. Yang, C. Hao, R. Li, F. Chang, Y. Wei, R. Tao, B. Dai, J. Yang, W. X. Zhao, and J.-R. Wen. ClawGym: A scalable framework for building effective Claw agents, 2026. URL <https://arxiv.org/abs/2604.26904>.
- [3] H. He, L. Wang, H. Chen, Y. Shao, Y. Shi, L. Wang, P. Liu, G. Lin, Z. Wang, H. Sun, T. Su, and C. Wan. CheckerBench: Can Long-Horizon Agents Synthesize Static-Analysis Checkers?, 2026. URL <https://arxiv.org/abs/2610.07557>.
- [4] H. Chen, Z. Hu, J. Chai, H. Yang, H. He, X. Wang, W. Lin, L. Wang, G. Yin, and Z. Zhao. ToolForge: A Data Synthesis Pipeline for Multi-Hop Search without Real-World APIs, 2025. URL <https://arxiv.org/abs/2512.16149>.
- [5] Z. Cheng, A. Mishra, S. Sahu, and W. Campbell. Efficient reinforcement learning for long-horizon tool-use agentic tasks, 2026. URL <https://arxiv.org/abs/2608.10357>.
- [6] C. Dong, C. Yue, H. He, Y. Liu, F. Tang, S. K. Zhou, X. Wang, J. Chai, and G. Yin. TAPO: Tool-Aware Policy Optimization via Credit Transfer for Multimodal Search Agents, 2026. URL <https://arxiv.org/abs/2606.05784>.
- [7] G. Dong, J. Lu, J. Huang, W. Zhong, L. Liu, S. Huang, Z. Li, Y. Zhao, X. Song, X. Li, J. Jin, Y. Zhu, H. Wang, F. Lei, Q. Luo, M. Chen, Z. Chen, J. Feng, J.-R. Wen, and Z. Dou. Agent-World: Scaling real-world environment synthesis for evolving general agent intelligence, 2026. URL <https://arxiv.org/abs/2604.18292>.
- [8] L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for LLM agent training, 2025. URL <https://arxiv.org/abs/2505.10978>.
- [9] S. Gandhi, S. Goyal, K. Kate, and Y. Rizk. DRACO: Fine-grained credit assignment with dynamic rubrics for long-horizon agent training, 2026. URL <https://arxiv.org/abs/2609.04094>.
- [10] B. Mao, H. He, L. Wang, L. Lin, M. Zhou, G. Liu, J. Liu, T. Huai, C. Zhang, B. Li, K. Lei, G. Dong, Z. Shao, R. Zheng, H. Yan, J. Zhou, C. Wan, T. Gui, L. He, and X. Qiu. WEFT: Scaling Tool-Use Post-Training for General-Purpose Agents, 2026. URL <https://arxiv.org/abs/2609.36887>.
- [11] Y. Hao, Z. Jin, H. Liao, K. Liu, and J. Zhao. Why multi-step tool-use reinforcement learning collapses and how supervisory signals fix it, 2026. URL <https://arxiv.org/abs/2606.26027>.
- [12] Z. Hua, Y. Yao, W. Xie, Y. Zhao, M. Liu, R. Qiu, Z. Huang, Z. Wang, Y. Ji, Y. Ye, L. Zhu, X. Lei, H. Li, Z. Ma, Z. Wang, Z. Zhang, and J. Liu. CLI-Universe: Towards verifiable task synthesis engine for terminal agents, 2026. URL <https://arxiv.org/abs/2606.22883>.
- [13] M. Ivanov and A. Rana. Anchor: Mitigating artifact drift in agent benchmark generation, 2026. URL <https://arxiv.org/abs/2605.26321>.
- [14] 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.
- [15] M. Li, E. Song, Y. Zuo, X. Liu, R. Wen, Q. Fu, G. Antichi, J. He, J. Tie, Z. Shao, X. Xue, X. Xiao, L. Zhong, S. Zhang, J. Zhao, J. Lu, S. Zhang, X. Sun, C. Zheng, Z. Fan, H. Li, T. Pan, X. Wu, Y. Song, X. Li, B. Lyu, M. Li, H. Dai, G. Chen, and S. Zhu. Scalable LLM Agent Tool Access in the Cloud, 2026. URL <https://arxiv.org/abs/2607.15593>.
- [16] Y. Li, Y. Feng, Z. Xu, Z. Ma, K. Zheng, F. Jiang, X. Sun, R. Shao, Z. Chen, Y. Huang, X. Han, B. Lee, K. Xu, S. Zeng, H. Hua, X. Zhang, B. Alomair, R. Krishna, L. Zettlemoyer, P. W. Koh, B. Ramasubramanian, L. Niu, X. Yue, and R. Poovendran. JobBench: Aligning agent work with human will, 2026. URL <https://arxiv.org/abs/2605.26329>.
- [17] H. He, C. Yue, C. Dong, M. Tian, H. Chen, Z. Liu, J. Chai, X. Wang, Y. Zhang, Q. Liao, G. Yin, W. Lin, C. Wan, H. Sun, and T. 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*, 2026. doi: 10.1145/3770855.3817466. URL <https://doi.org/10.1145/3770855.3817466>.
- [18] H. Chen, S. Zhou, Z. Hu, J. Chai, H. He, H. Yang, X. Wang, W. Lin, T. Wang, Z. Zhao, and G. Yin. ToolForge: A Data Synthesis Pipeline for Multi-Hop, Multi-Turn, and Self-Reflective Tool-Use Data. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. URL <https://openreview.net/forum?id=vFFeh066Y0>.
- [19] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999. URL <https://ai.stanford.edu/~ang/papers/shaping-icml99.pdf>.

- [20] Z. Niu, Y. Zhao, J. Zhang, X. Shen, J. Si, H. Tian, C. Xu, Y. Song, J. Mao, Y. Gao, Y. Li, J. Wu, L. Yan, S. Wang, and D. Yin. DuMateBench: Evaluating autonomous agents in complex real-world workflows, 2026. URL <https://arxiv.org/abs/2608.26546>.
- [21] K. Opsahl-Ong, A. Singhvi, J. Collins, I. Zhou, C. Wang, A. Baheti, O. Oertell, J. Portes, S. Havens, E. Elsen, M. Bendersky, M. Zaharia, and X. Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL <https://arxiv.org/abs/2603.08655>.
- [22] Z. Pan, J. He, K. Zhang, Y. Han, Z. Liu, Y. Zhao, Y. Ye, and Z. Yang. Meta-Task: Turning terminal task synthesis into a terminal task for scalable agent training, 2026. URL <https://arxiv.org/abs/2607.27929>.
- [23] Y. Pandya, S. Gupta, S. Harne, A. Yadav, K. Chourasia, H. Mozannar, V. Vineet, S. Abdali, C. Rosset, Y. Lara, A. Awadallah, E. Kamar, and A. Nambi. Echoverse: Deep, evolving environments for training computer-use agents at scale, 2026. URL <https://arxiv.org/abs/2607.28074>.
- [24] J. Peng, Y. Liu, R. Zhou, C. Fleming, Z. Wang, A. Garcia, and M. Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents, 2026. URL <https://arxiv.org/abs/2602.16165>.
- [25] Qwen Team. Qwen3.5: Towards native multimodal agents, Feb. 2026. URL <https://qwen.ai/blog?id=qwen3.5>. Official citation recommended by the Qwen3.5-35B-A3B [model card](#); accessed September 11, 2026.
- [26] J. Chai, G. Yin, Z. Xu, C. Yue, Y. Jia, S. Xia, X. Wang, J. Jiang, X. Li, C. Dong, H. He, and W. 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>.
- [27] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- [28] D. Shepard and R. Salimans. AutomationBench, 2026. URL <https://arxiv.org/abs/2604.18934>.
- [29] K. Shi, Z. Wang, Q. Su, S. Huang, Z. Zhang, Z. Fang, Q. Ren, J. Liu, Y. Zeng, Y. Zhao, L. Chen, Z. Chen, and F. Zhao. FACET: Preserving source intent and executable state in terminal task synthesis, 2026. URL <https://arxiv.org/abs/2608.18580>.
- [30] X. Song, H. Chang, G. Dong, Y. Zhu, J.-R. Wen, and Z. Dou. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis, 2026. URL <https://arxiv.org/abs/2601.05808>.
- [31] J. Su, Z. Zhao, H. Wang, and H. Chen. CoAL-RAG: A Complexity-Aware Legal Retrieval-Augmented Generation Method, 2026. URL <https://arxiv.org/abs/2608.17536>.
- [32] X. Sun, Z. Shao, X. Shi, S. Xiao, C. Ma, X. Xue, J. Lu, S. Zhang, E. Song, S. Yang, X. Li, C. Fang, C. Fang, B. Lyu, and S. Zhu. Enabling Black-box RPC-API Testing with Multi-Agent Reinforcement Learning and LLMs: An Industry Case Study. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, pages 81–93. ACM, 2026. doi: 10.1145/3786583.3786855. URL <https://doi.org/10.1145/3786583.3786855>.
- [33] Z. Tang, X. Zhou, Y. Liu, L. Li, Y. Wu, W. Wang, H. Huang, W. Zhou, J. Zhou, J. Song, S. Yu, J. Wang, Z. Zhou, H. Zhou, Y. Lv, J. Li, J. Liu, R. Chen, C. Liu, G. Li, J. Kang, and F. Wu. Workspace-Bench 1.0: Benchmarking AI agents on workspace tasks with large-scale file dependencies, 2026. URL <https://arxiv.org/abs/2605.03596>.
- [34] L. Tao, B. Peng, W. Yao, T. Ge, H. Cheng, M. H. Wang, J. Gao, and S. Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL <https://arxiv.org/abs/2607.13988>.
- [35] Tencent WorkBuddy Bench Team. Tencent WorkBuddy Bench: A multi-domain coding-agent benchmark with contamination-resistant task construction, 2026. URL <https://arxiv.org/abs/2607.20911>.
- [36] H. Trivedi, T. Khot, M. Hartmann, R. Manku, V. Dong, E. Li, S. Gupta, A. Sabharwal, and N. Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents, 2024. URL <https://arxiv.org/abs/2407.18901>.
- [37] H. He, C. Yue, C. Dong, C. Wan, T. Su, H. Sun, J. Chai, X. Wang, and G. Yin. VistaHop: Benchmarking Long-Horizon Visual DeepSearch, 2026. URL <https://arxiv.org/abs/2606.03273>.
- [38] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *Proceedings of the 31st International Conference on Software Engineering*, pages 364–374. IEEE, 2009. doi: 10.1109/ICSE.2009.5070536. URL <https://profsforrest.github.io/homepage/data/publications/2009-icse-genprog.pdf>.
- [39] X. Wu, C. Yang, H. Liu, X. Lin, Z. Shi, H. Zhou, X. Jiang, C. Xu, J. Li, and J. Guo. Envs-FORGE: Frontier-optimized reward-grounded environment synthesis for agent RL, 2026. URL <https://arxiv.org/abs/2608.14312>.
- [40] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, X. Guo, D. Yang, C. Liao, W. He, S. Gao, L. Chen, R. Zheng, Y. Zou, T. Gui, Q. Zhang, X. Qiu, X. Huang, Z. Wu, and Y.-G. 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)*, pages 27914–27961. Association for

- Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1355. URL <https://aclanthology.org/2025.acl-long.1355/>.
- [41] Z. Xi, J. Huang, C. Liao, B. Huang, H. Guo, J. Liu, R. Zheng, J. Ye, J. Zhang, W. Chen, W. He, Y. Ding, G. Li, Z. Chen, Z. Du, X. Yao, Y. Xu, J. Chen, T. Gui, Z. Wu, Q. Zhang, X. Huang, and Y.-G. Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL <https://arxiv.org/abs/2509.08755>.
 - [42] Z. Xi, C. Liao, G. Li, Y. Yang, W. Chen, Z. Zhang, B. Wang, S. Jin, Y. Zhou, J. Guan, W. Wu, T. Ji, T. Gui, Q. Zhang, and X. Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress, 2025. URL <https://arxiv.org/abs/2511.08325>.
 - [43] K. Xu, H. Xu, X. Chen, Y. Wang, Z. Li, X. Liu, C. Wu, J. Xia, and Y. Li. STAMP: Provenance-guided credit assignment for deep search agents, 2026. URL <https://arxiv.org/abs/2607.11172>.
 - [44] C. Yue, C. Dong, Y. Gao, H. He, J. Chai, W. Lin, and G. Yin. Promoting Efficient Reasoning with Verifiable Stepwise Reward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34530–34538, 2026. doi: 10.1609/aaai.v40i41.40752. URL <https://doi.org/10.1609/aaai.v40i41.40752>.
 - [45] W. Zeng, J. Zhang, H. Chen, Z. Hu, Y. Liang, J. Chai, D. Liu, Z. Liu, S. Yan, M. Xue, X. Wang, W. Lin, and G. Yin. RecRM-Bench: Benchmarking Multidimensional Reward Modeling for Agentic Recommender Systems, 2026. URL <https://arxiv.org/abs/2605.11874>.
 - [46] H. Zhang, M. Chen, D. Zhou, C. Lv, H. Chang, S. Cui, F. Wu, and S. Zhou. TRCA: Transition-wise rubric credit assignment for long-horizon LLM agents, 2026. URL <https://arxiv.org/abs/2608.16156>.
 - [47] Z. Zhong, I. Segal, I. Bercovich, S. Saxena, K. Zhang, and A. Raghunathan. Hardening agent benchmarks with adversarial hacker-fixer loops, 2026. URL <https://arxiv.org/abs/2606.08960>.
 - [48] C. Zhou, Q. Jiang, S. Wu, and X. Zhou. Coverage, not targeting: A structural regime in multi-turn agent credit assignment, 2026. URL <https://arxiv.org/abs/2609.02417>.
 - [49] S. Zhou, H. Chen, T. Wang, G. Lin, Z. Wang, and H. Zhao. RAG-Stress: Probing the Limits of Evidence Reliance in Retrieval-Augmented Generation, 2026. URL <https://arxiv.org/abs/2610.11183>.

A Extended Related Work

Executable environments and task synthesis. Professional agent training requires executable tasks, recoverable state, and observable outcomes across related artifacts. AppWorld provides controllable application environments; EnvScaler supports programmatic environment construction, while ClawGym materializes workspaces around task and capability requirements [2, 30, 36]. Agent-World uses capability diagnostics to expand tool/database environments [7]. ToolForge constructs virtual-tool tasks, and MetaTask and CLI-Universe emphasize executable validation [4, 12, 18, 22]. These approaches organize synthesis around task bundles, environment expansion, or capability requirements. WORKROVER grounds task construction in file relations and professional skills, then assembles Tasks and Verifiers within recoverable Work Contexts under a versioned Harness. Shared contexts also make dependency tracking and revalidation necessary when one component changes.

Grounded acceptance and verifier reliability. Acceptance checks must agree with the instruction and the evidence available in the workspace. ANCHOR constructs instructions, environments, solutions, and verifiers from a shared workflow specification [13]. FACET grounds artifacts in shared state and repairs inconsistencies while preserving source intent [29]. Hacker-Fixer audits tests and environments with adversarial cases while checking legitimate solutions [47]. Stronger rejection checks therefore need to be considered alongside false rejection of valid alternatives. WORKROVER freezes instruction-grounded requirements and criterion-evidence bindings, then evaluates implementations using violating mutants and independently valid solutions. Reference artifacts supply recomputable anchors without adding requirements. Infrastructure-invalid episodes are distinguished from policy failures. Validation remains limited to the declared Contract and the coverage of the controls.

Environment evolution and repair selection. Envs-FORGE studies joint revision of instructions, fixtures, solutions, tests, and environments, while EchoVerse studies executable environment evolution [23, 39]. FACET routes failed validation to implicated artifacts and repeats validation from a clean state [29]. Program repair likewise studies patch minimization: GenProg reduces test-passing patches through structural differencing and delta debugging [38]. FCR freezes component-level proposals over Context, Task, Harness, and Verifier before replaying their combinations. Materialization includes dependency-induced changes, and promotion checks affected retained tasks for regressions under fixed requirements. The selected repair is the smallest admissible closed edit set among the tested combinations; bounded enumeration does not establish a global minimum over all possible repairs. This separates proposal generation, compatibility selection, and validation of a shared environment revision.

Local progress in long-horizon learning. Methods differ in the evidence used to assign local feedback. GiGPO uses state comparisons, while HiPER uses a planning hierarchy [8, 24]. TRACE forms temporal differences in frozen-model reference-answer likelihoods, with bounded look-ahead for delayed tool effects [34]. TRCA combines deterministic transition-rubric scores with increments in cumulative condition coverage [46]. VTCA instead evaluates successive observable workspace states with frozen checkers. Signed progress can register both completion and setbacks in persistent, mutable artifacts. Unsupported or unavailable evidence supplies no local signal, while terminal evaluation remains applicable to supported task outcomes. These differences concern the feedback source and its coverage; they do not by themselves establish the comparative effectiveness of individual components.

Evidence attribution and shaping limits. STAMP maps verified citations to first-exposure actions; DRACO uses judge-attributed steps with sign-preserving GRPO advantage redistribution [9, 43]. VTCA follows recorded direct use of exact tool outputs or artifact versions, assigning at most half of positive progress to eligible prerequisite turns and leaving negative increments local. Access alone creates no dependency edge. Incorrect provenance can misallocate credit, and missed links leave prerequisites unrepresented. Recorded use therefore does not establish causal necessity. Potential-based reward shaping preserves optimal policies under stated assumptions [19]; VTCA is formulated as workspace-grounded advantage shaping. Conservation

of checker increments before gating, costs, and clipping establishes neither policy invariance nor an unbiased policy-gradient estimate.

B Data Construction and Case Studies

B.1 Work Context Construction

For each source workspace w , the index $I(w)$ stores a typed graph $G_w = (V_w, E_w)$ with files or versions as nodes and semantic, structural, temporal, provenance, and workflow edges. Directory co-location is only a weak structural signal. A job skill specifies its professional setting, workflow, materials, and deliverables. The Skill Matcher retrieves related file regions, then expands them through file and application skills with compatible inputs, outputs, and interfaces. The Harness validates dependencies, permissions, bindings, reset, recovery, and evidence capture. Recorded lineage and derived task branches stay in one data split.

Construction complexity. Let J be the number of job skills, W the number of source workspaces, 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-job-skill 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 Work Context construction is

$$O(WC_{\text{index}}) + O(JC_{\text{retrieve}}) + O(Jk(C_{\text{agent}} + C_{\text{match}})) + O(MC_{\text{env}}). \quad (2)$$

Agent inspection covers the Jk shortlisted candidates, with cost $O(JkC_{\text{agent}})$, whereas exhaustive job-workspace inspection incurs $O(JWC_{\text{agent}})$. This comparison concerns agent inspection; indexing, retrieval, skill matching, and environment assembly remain separate costs. The index is reusable across job-skill queries, so its construction cost is paid once for a fixed source collection.

B.2 Synthesis Resources and Coverage

The inventory contains 21,444 app skills, 5,918 file skills, and 36,064 job skills covering 317 roles. App and job skills receive one category label across 15 and 10 categories; file skills have 11,273 memberships across 22 categories. Figure 4 shows the six largest categories per library and groups the remainder as Other. Construction profiles describe artifact diversity, file relations, workflow horizon, cross-root dependencies, and noise; empirical difficulty is calibrated separately.

B.3 Workspace Construction from Deep Research Sessions

In this route, a source workspace is the exported file collection associated with one research record. Historical research supplies materials; new tasks require fresh Demonstrator execution and independent admission (Figure 8).

Inventory and source identity. Files are inventoried by relative path, format, size, and content hash. Bounded previews expose headings, workbook sheets, or archive members for selective reading. Captured sources and independently retained, readable notes, tables, or reports may enter the Starter State after relevance and leakage checks. Raw conversations, tool logs, internal analysis caches, provenance directories, and unsupported web caches are excluded. The original query informs construction only and is not Solver evidence. Source pointers support audit; a URL cannot replace a required local file. Citations in notes must be checked against their sources. Co-occurrence, retrieval order, and model assertions do not establish dependencies; missing lineage is unresolved. Freezing files does not make historical searches or services replayable. Unavailable required sources lead to revision or rejection; permitted new retrieval follows the Harness contract and records its evidence separately.

Branch plans and answer leakage. A job skill’s requirements guide eligible-file retrieval within one source collection. File skills bind the required operations; app skills bind any instrumented application capabilities. Branches must differ in professional decisions, operations, or principal products, and cannot import unselected files or combine source collections. A feasible Branch Plan becomes a workspace and Task Blueprint specifying

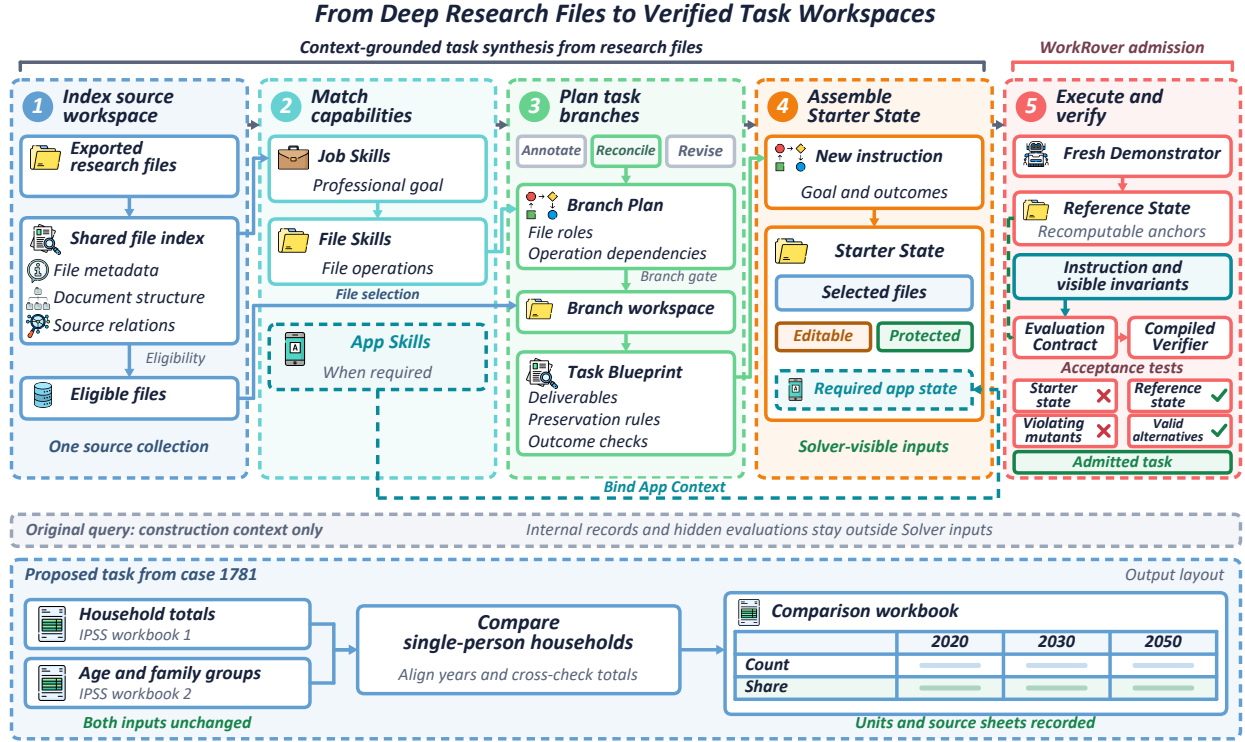


Figure 8 From Deep Research files to verified task workspaces. Instructions and visible invariants define requirements; the dashed green arrow supplies recomputable Reference State anchors. Case 1781 uses two IPSS workbooks; shaded cells mark output values to populate.

file roles, dependencies, deliverables, and modification/preservation constraints. Every required outcome has an observable criterion. A write acknowledgment or readback excerpt alone is not an editable artifact. Retained notes require evidence for their claims; inputs that already complete the new task cause revision or rejection. Hidden plans, solution keys, and evaluation records stay outside Solver inputs and cannot supply unstated requirements or inaccessible task evidence.

Fresh execution, verification, and lineage. The instruction and selected files/application state define a frozen Starter State. A fresh Demonstrator establishes feasibility and a Reference State; historical session completion cannot substitute. The Contract uses only the instruction and visible invariants, with independently recomputable anchors. Admission requires rejection of the Starter State and violating mutants, acceptance of valid Reference States and alternatives, and protected-state checks. Branches retain collection, session, and file-version lineage. Sessions, retained collections, candidates, and admitted tasks are counted separately; branch count alone does not establish independent diversity.

B.4 Task Synthesis and Quality Control

Each package contains an instruction, resettable snapshot, environment specification, metadata, and a fresh Demonstrator realization with available terminal artifacts. Instructions state outcomes and constraints without prescribing tools or action sequences. Core files require defensible relations; input ablation removes unnecessary bindings, and distractors cannot change the dependency closure. Table 4 controls scale and interaction horizon.

Empirical difficulty is the valid-rollout failure rate of a frozen reference panel under matched snapshots, tools, permissions, and budgets. Panel membership, aggregation, and band thresholds are fixed on development data; evaluated systems do not contribute to these labels. Gates check instruction clarity, readable inputs, sufficient dependencies, reproducible restoration, and outcome anchors. Unobservable core outcomes are rejected. External capabilities require declared Harness tools with recorded versions, seeds, permissions, and

Table 4 Target task-construction profiles. The profiles control scale and interaction horizon.

Dimension	P1	P2	P3
Edit scope (files)	~ 10	~ 20	> 30
Demonstrator horizon target (turns)	5–10	15–30	> 30
Skill composition target	2 job + ≥ 1 file	3 job + ≥ 3 file	4 job + ≥ 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

initial-state identities.

B.4.1 Task Design and Review

The Blueprint binds each outcome to extractable or independently recomputable artifact/application evidence (Table 5). The manifest distinguishes required inputs, history, distractors, and protected artifacts. Review checks opening/parsing, formulas, merged cells, annotations, version history, layout, and embedded objects; visually specified outcomes also require rendered inspection.

Table 5 Representative workflow requirements and observable outcome evidence.

Workflow	Inputs and dependencies	Outcome evidence
Artifact revision	Source versions, comments, approvals	Required edits, consistent versions, preserved review history/protected content
Data reconciliation	Datasets, formulas, source records, periods	Recomputed aggregates, formulas, units, declared missing-value handling
Structured content	Reports, templates, charts, media	Required content/layout and preserved unaffected elements
Cross-artifact delivery	Linked files, output specifications, exports	Complete outputs, formats/order, source-export consistency
Collaboration	Calendar, email, drive, approval state	Required state changes, consistent records, satisfied workflow conditions

B.5 Case Studies: 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.3. 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.6 Case Study: Quarterly Executive Handoff

Record 0805:comp_018554 belongs to the 5,000-trajectory pool and the 3,200-trace subset described in Appendix D.1. Figure 11 follows its Hermes execution. The agent verifies five regional revenue rows, registers the vendor, amends the Q2 access policy, retires the archived Q1 policy, and checks five exports. Revenue values match the source; the workbook adds totals, growth, and evidence references. The final report retains the database lock wait and stale export alias as follow-ups. All four archived goal groups pass (score 0.946); file validation covers structure/content, with visual review unavailable.

C Verifier, Repair, and Training Details

Case 1875 Prompt-engineering course design

118 messages

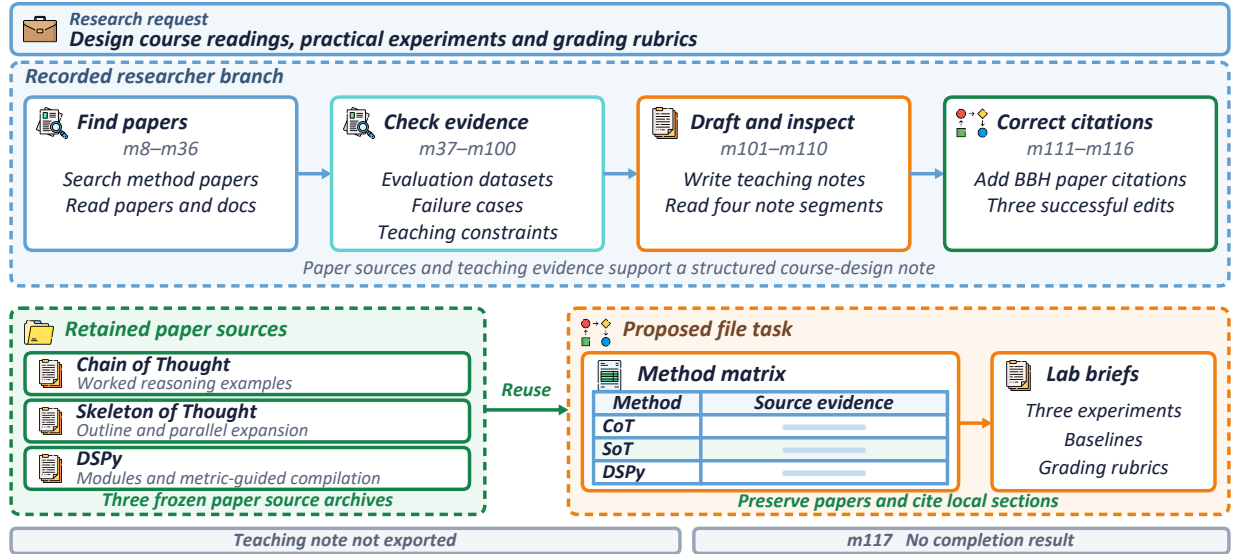


Figure 9 Recorded example 1875: prompt-engineering course design. This researcher branch writes `teaching_-design.md`, inspects four segments, and adds the BIG-Bench Hard paper citation in three places. The proposed task reuses captured CoT, SoT, and DSPy paper-source archives to produce a method comparison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

Case 1781 Policy research on older adults living alone

69 messages

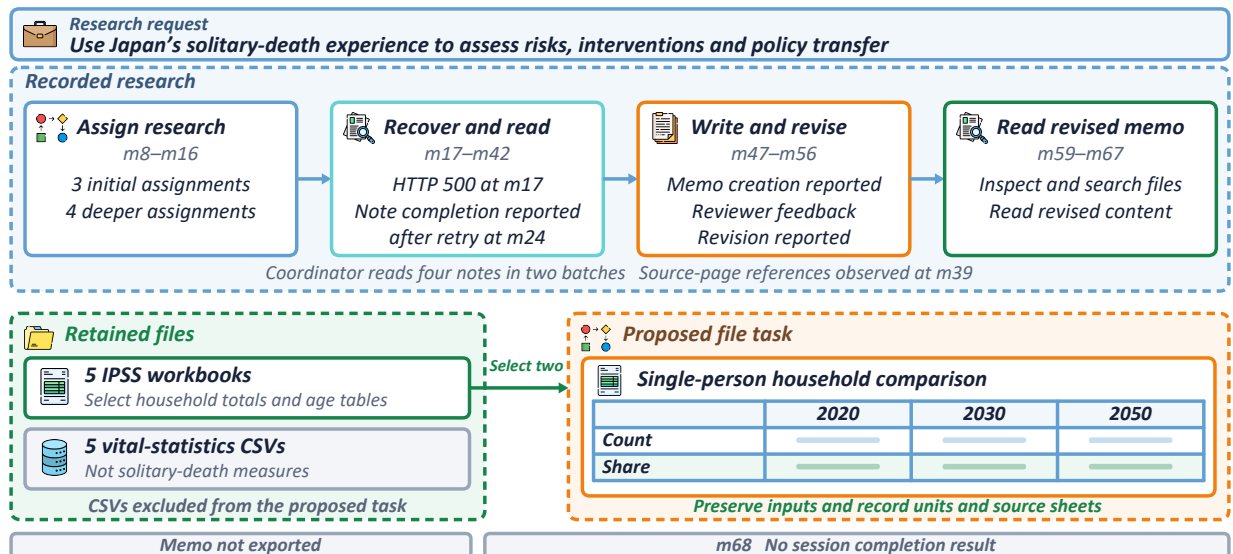


Figure 10 Recorded example 1781: policy research on older adults living alone. The proposed task selects `hprj2024_gaiyo_kekka1.xlsx` and `hprj2024_gaiyo_kekka2.xlsx`, retaining category definitions, units, and source-sheet references. The revised `policy_memo.md` is observed through tool readbacks. Source-page references at `m39` cite IPSS and MHLW.

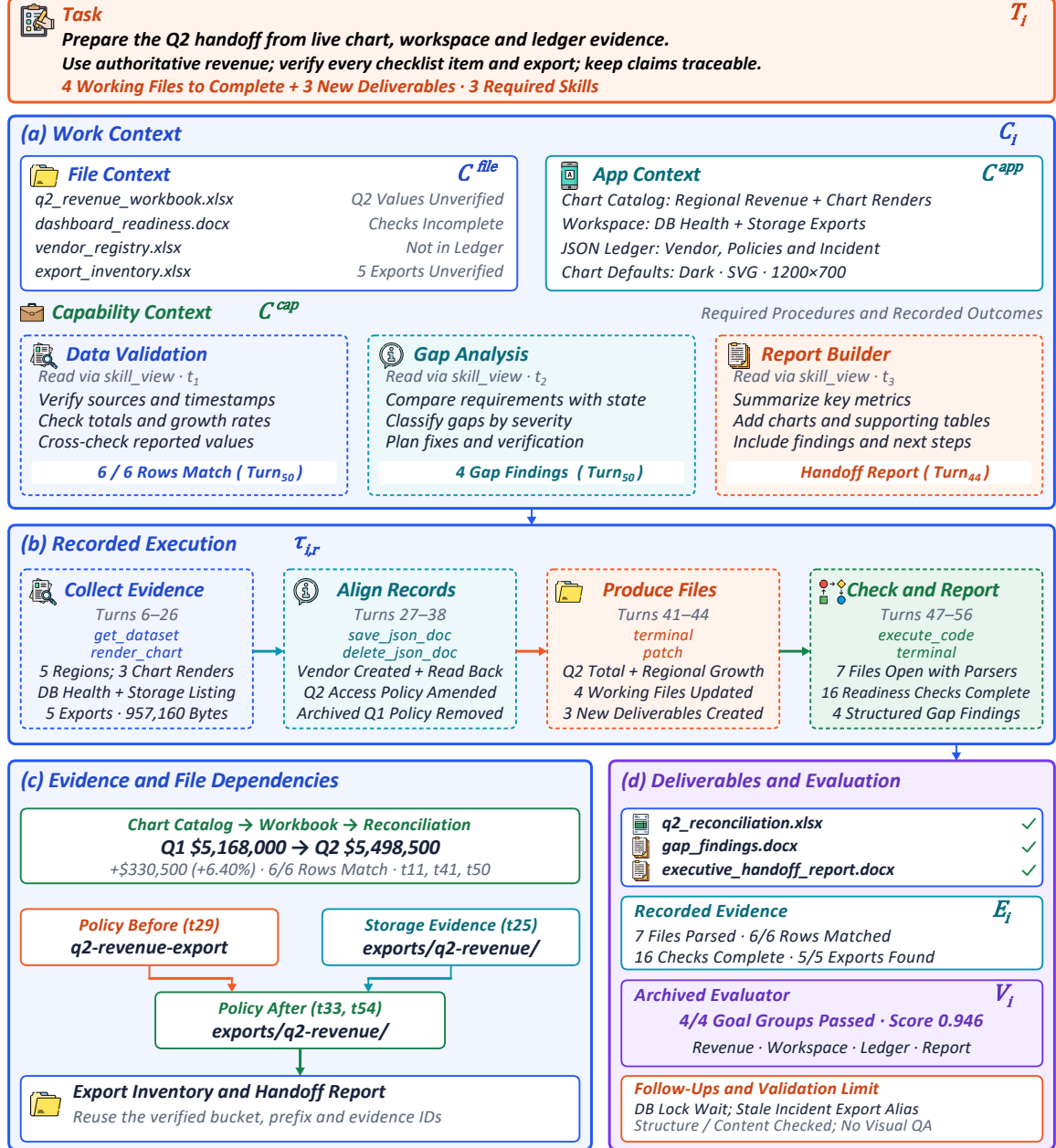


Figure 11 Recorded quarterly executive handoff. (a) File, capability, and app context. (b) Selected rollout stages. (c) Revenue and export-path evidence supporting consistent updates. (d) Deliverables, archived evaluation, retained follow-ups, and validation scope.

C.1 Verifier Specification and Calibration

Criteria derive from the instruction, Starter State, and visible invariants; Reference artifacts only instantiate independently recomputable anchors. The frozen Contract specifies editable/protected state, allowed transitions, and each criterion’s predicate, evidence, checking mode, weight w_k , threshold γ_k , and critical flag b_k . Unresolved requirements prevent admission. Adapters and canonical diffs provide stable evidence IDs. Executable checks read both snapshots; at most one batched Judge call receives frozen criterion-tagged evidence, has no live tools, and cites evidence or abstains. For scores $p_{i,k} \in [0, 1]$, set $y_{i,k} = \mathbf{1}[p_{i,k} \geq \gamma_{i,k}]$ and

$$\bar{w}_{i,k} = w_{i,k} / \sum_j w_{i,j}; \quad S_{B,i} = \sum_k \bar{w}_{i,k} p_{i,k}, \quad H_i = g_i^{\text{integrity}} g_i^{\text{protected}} g_i^{\text{safety}}, \quad G_i = H_i \prod_{k: b_{i,k}=1} y_{i,k}. \quad (3)$$

Healthy evaluation passes iff $G_i = 1$ and $S_{B,i} \geq \tau_B$. Missing required state implies failure; infrastructure-unavailable evidence implies invalidity. Policy errors under healthy infrastructure are not invalid. Partial credit cannot offset gates. Auxiliary $S_A \in [0, 1]$ uses frozen trace checks. Admission requires the Starter–Reference contrast, protected-state checks, no unresolved invalid controls, and frozen thresholds for criterion agreement, worst-family mutant detection, and valid-alternative acceptance. Task-level negatives must violate the Contract. Calibration, initial admission, reporting, and promotion use disjoint recipe–seed banks. Unsupported semantic criteria retain terminal supervision without local progress signals.

C.2 FCR Mechanics and Promotion

Freeze one direct proposal per available Context/Task/Harness/Verifier factor before replay: at most 16 cells, or eight before compilation. Each cell χ applies the dependency-closed update set $\Gamma(\chi)$, including dependency-induced updates; changed tasks require fresh execution/compilation, and shared Harness edits require affected-task checks. Established conflicts make cells unavailable; uncertain cells proceed to materialization. Candidate repair returns to the earliest affected gate; maintenance freezes the admitted parent. Both preserve intent, outcomes, transitions, and applicable Contract weights, thresholds, flags, and gates. Hidden evidence cannot leak into visible edits; new tools require separate releases and semantic changes create new candidates. Blinded equivalence review and a frozen solver panel apply development-fixed simplification margins.

Matched Context–Task–Harness seeds produce evidence reused by Verifier variants. Health uses all seeds; outcome contrasts use common healthy seeds on complete compatible subcubes, without imputation. Verifier contrasts use sealed label agreement. Selection requires a proposal-independent defect/health predicate, proposal-disjoint replay, equivalence, no affected-task regression, and compiled-Verifier calibration plus a one-use binary promotion shard. Reporting audits never select repairs. The smallest validated closed edit set $\Gamma(\chi^*)$ among tested cells receives fresh-seed confirmation (Section 3.2); failure discards candidates or retains maintenance parents and quarantines unresolved triggers.

C.3 FCR Evaluation Protocol

Frozen stratified screening of the 10K pool targets 300 natural mismatches, classified by two blinded reviewers with third-reviewer adjudication. The separate controlled cohort injects 260 concealed Contract-preserving defects: 80 single-, 120 pair-, 40 triple-, and 20 four-factor cases. Candidate-controlled comparisons share proposals; end-to-end comparisons allow independent proposals under matched caps. Each distinct Context–Task–Harness configuration uses five matched seeds; compatible Verifiers reuse evidence. Confirmation seeds and generation-specific promotion shards are disjoint. Joint necessity excludes passing strict proposal subsets. Triggers are statistical units; seeds/cells are nested measurements.

C.4 Verifier-to-Turn Credit Assignment

Canonical traces retain ordering, call/result IDs, tool definitions, versions, and state metadata. SFT supervises agent tokens; invalid rollouts receive no training signal. For valid rollout r ,

$$R_{i,r} = G_{i,r}[(1 - \lambda)S_{B,i,r} + \lambda S_{A,i,r}], \quad A_{i,r}^{\text{traj}} = \frac{R_{i,r} - \bar{R}_i}{\max(\text{sd}_{r'} R_{i,r'}, \sigma_{R,\min})}, \quad 0 \leq \lambda \leq 0.1. \quad (4)$$

The denominator floor is development-fixed. Deterministic progress $\Delta_{i,r,t,k} = \phi_{i,k}(x_{i,r,t+1}) - \phi_{i,k}(x_{i,r,t})$ requires both endpoints evaluable; otherwise record zero increment and a missing reason. Missing intervals are not bridged or treated as zero completion. No intermediate Judge is called. Prerequisite set $\mathcal{P}_{i,r,s,k}$ deduplicates earlier producer turns whose exact outputs/versions are directly consumed at s , with criterion-compatible immutable IDs and call/result links. Stale, ambiguous, inferred, and unrecorded uses are excluded.

With $\rho_{i,r,s,k} = \rho \mathbf{1}[|\mathcal{P}_{i,r,s,k}| > 0]$ and $0 \leq \rho \leq 1/2$,

$$b_{i,r,s,t,k} = \begin{cases} 1 - \rho_{i,r,s,k}, & t = s, \\ \rho_{i,r,s,k}/|\mathcal{P}_{i,r,s,k}|, & t \in \mathcal{P}_{i,r,s,k}, \\ 0, & \text{otherwise,} \end{cases} \quad (5)$$

$$\delta_{i,r,t}^{\text{crit}} = \sum_k \bar{w}_{i,k} \left[\sum_{s=t}^{T_{i,r}-1} b_{i,r,s,t,k} \max(\Delta_{i,r,s,k}, 0) + \min(\Delta_{i,r,t,k}, 0) \right].$$

Allocation is direct; negatives and empty-set increments stay local. Weights are not renormalized over observed criteria or centered across turns. Equation 1 gates progress by H , terminal reward by G . Trace cost $c_t \in [0, 1]$ covers malformed/error calls, redundant state-preserving actions, and attributable protected-state/safety violations. GRPO averages tokens within turns, turns within trajectories, then trajectories within task groups, using fixed behavior policy, SFT reference, and detached advantages. Before gating/-cost/clipping, $\sum_t \delta_{i,r,t}^{\text{crit}} = \sum_k \bar{w}_{i,k} \sum_t \Delta_{i,r,t,k}$; only complete observations telescope. Write-revert cycles and wrong provenance can still alter updates.

D Evaluation Details Referenced in the Main Text

D.1 Experimental Setup

Training and visualization data. Qwen3.5-35B-A3B is the post-training backbone; 397B-A17B supplies 30,115 Verifier-accepted SFT trajectories passing structural/lineage QA. Collection allows three retries per rejected turn with exact prefix restoration. Four 35B-A3B pilots per candidate retain 3,200 RL pairs from 20,000 candidates; splits are by Work Context. RL starts from SFT with turn-balanced GRPO, KL coefficient 0.01, and matched tools, permissions, states, and budgets. Trajectory analysis separately selects 3,200 of 5,000 successful traces: $40 < \text{deduplicated assistant turns} < 100$, $\text{eval.score} > 0.8$, passed evaluation, complete segments, and normal termination. Ranking uses named skill calls, score, then deterministic hash. This subset does not estimate online RL; valid failures still train. The Harness allocation is OpenCode/Claude Code/Codex/Hermes 60/20/10/10%.

Controlled evaluations. Matched runs fix release, split, evaluator, Harness, and budget, charging invalid/retried attempts under frozen caps. Tasks cluster by Work Context. GDPval Elo and percentages are not averaged. Controlled OfficeQA Pro uses 133 questions at 0.0% numerical tolerance.

Benchmark configuration. GDPval v2 uses the September 12, 2026 (Asia/Shanghai) AA snapshot and Stirrup [1]. JobBench Main uses 65 tasks, OpenCode v1.14.18, and Grok 4.3 [14]. For OfficeQA Pro, WORKROVER uses Claude Code, temperature 0.7, top- p 0.95, top- k 40, 1,800 seconds.

D.2 Matched VTCA Comparison

Figure 6 matches checkpoints, task/seed schedules, groups, rollout/token/update budgets, and interaction/Verifier/Judge caps; all attempts are charged. VTCA uses one terminal Verifier pass and at most one Judge call; checker/capture/CPU costs are separate. Controls cover aggregation, local-only progress ($\rho = \eta = 0$), matched random/verified recipients, and trace cost. Random recipients preserve producer mass, budget, and count. Inference resamples training replicas and task/Context clusters; GDPval refits ratings from matched judgments.

E Data Construction and Case Study Summaries

The following tables consolidate the material boundaries and admission process in Appendices B.3 and B.4, and distinguish proposed tasks from the recorded execution in Appendices B.5 and B.6.

Table 6 Source materials, construction roles, and Solver visibility. Eligibility does not establish factual authority or evidence sufficiency.

Material	Construction role	Solver visibility
Captured source files	Readable source documents, code, or tables.	Selected files are frozen in the Starter State.
Retained working files	Editable files; factual claims require supporting sources.	May be selected if they do not reveal the completed answer.
Original research query	Background for job skills and task intent.	Construction only; never Solver input or evidence.
Logs and internal analysis	Audit context; readbacks are not editable artifacts.	Excluded from selectable inputs.
Provenance and source pointers	Lineage/source identity; URLs cannot replace local files.	Internal manifests remain outside Solver inputs.

Table 7 Task construction stages and admission checks. Each new task requires fresh execution and verification, independently of historical session completion.

Stage	Recorded output	Acceptance check
Index the source	Paths, formats, sizes, hashes, structural previews, and available file relations.	Selected inputs are readable; claimed dependencies have support and unresolved lineage is recorded.
Plan a task branch	Job/file/app skill bindings and a Branch Plan selecting files from one collection.	The objective is feasible; branches differ in decisions, operations, or products; no unselected files are imported.
Assemble the Starter State	Frozen files and app state, permissions, capability bindings, and a role/protection manifest.	Dependencies are available; reset/recovery works; task-relevant structures survive; no completed-answer leakage.
Specify and execute	Instruction, Task Blueprint, fresh Demonstrator trace, and Reference State.	Required outcomes and preservation constraints are observable and achievable under the declared Harness.
Compile and calibrate	Frozen Contract, recomputable anchors, evidence bindings, and counterfactual controls.	The Starter State fails; valid Reference States and alternatives pass; violating mutants and protected-state changes are detected.
Record lineage and split	Source/session/file-version identities and separate collection, candidate, and admission counts.	Connected lineage and derived branches remain in one split; branch count is not treated as independent diversity.

Table 8 Case study inputs, work products, and evidence scope. The two Deep Research cases illustrate proposed derivative tasks; the handoff case is a recorded execution.

Case	Inputs and requested work products	Evidence and verification scope
1875: course design	Captured CoT, SoT, and DSPy paper archives support a method comparison workbook and three lab briefs with baselines and grading rubrics.	Claims cite local source sections; archives remain unchanged. The historical teaching note was not exported. Fresh task execution and admission are still required.
1781: household projections	Two IPSS workbooks support a comparison of single-person household counts/shares for 2020, 2030, and 2050, retaining definitions, units, and source-sheet references.	Both input workbooks remain unchanged. Historical memo evidence is limited to tool readbacks; source-page references do not prove individual downloads. Fresh admission is required.
Quarterly executive handoff	Revenue sources, working files, chart/-workspace state, and the ledger support updated files, a reconciliation workbook, gap findings, and an executive handoff report.	Record 0805:comp_018554: all four goal groups passed, score 0.946. Structure/content were checked; visual review was unavailable. Lock-wait and stale-alias follow-ups remain.