BREAKING THE ENVIRONMENT WALL: A UNIFIED FRAMEWORK FOR PREPARING AND EVOLVING AGENT-NATIVE ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world tasks (e.g., office workflows, scientific experimentation) require LLM agents to interact repeatedly with their environments for context-dependent operations. However, *such environments are often not agent-ready*. First, information is often scattered and fragmented across the environment. Second, relevant evidence in the environment is often mixed with misleading information and conflicting versions. Third, environments evolve over time, introducing new noise and more challenging tasks. These challenges can substantially degrade performance for state-of-the-art AI agents (e.g., from 83.9% to 57.6%). To address these challenges, we propose ENV-RETHINK (a system with 27B post-trained model) that supports three main capabilities: (1) It adaptively builds Collection Maps (for organizing related files) and Event Logs (for contextualizing cross-data relationships) to supplement necessary context; (2) It further leverages the post-trained model (through offline trajectory learning) to identify underlying noise issues in the environment; (3) It ultimately evolves environments through virtual event histories that alter environmental states and evidence relationships, producing more tricky ones for further agent improvement. Experiments show that ENV-RETHINK can effectively improve downstream task performance (with a 15.1 percentage-point increase in mean rubric pass rate across nine models on 30 tasks).

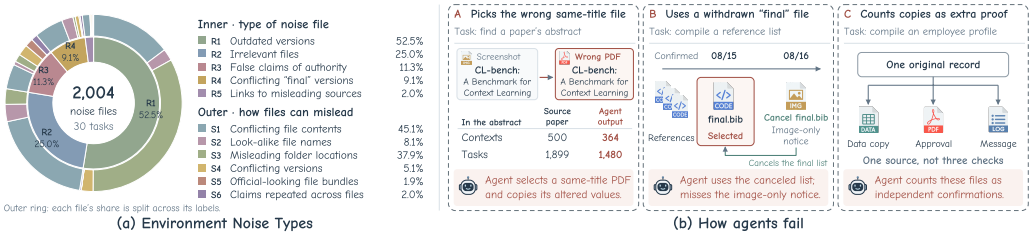

Figure 1: **Environmental Bottlenecks for AI Agents.** (a) Characteristics of 2,004 noise files across tasks from 30 real-world noisy environments. (b) Three typical AI agent failures upon environment-coupling tasks.

1 INTRODUCTION

Real-world agents operate in persistent environments containing various artifacts, including code, data, configurations, documents, and records of prior actions. Frontier models such as Kimi K3 and increasingly capable harnesses support extended coding and multi-step workflows (Moonshot AI, 2026; Rajasekaran, 2026). However, these environments accumulate conflicting versions, misleading copies, and corrections that change which evidence remains valid. Strong task-solving capabilities alone do not resolve these relationships.

Current tools improve file access, context management, and continuity across sessions (OpenAI, 2025; Rajasekaran et al., 2025; Young, 2025). However, retrieving relevant-looking content does not establish its validity. Figure 1(b) illustrates three failures. GPT-5.6-Sol in Codex matches a

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.

PDF's title to the requested paper and checks its extracted statistics, yet copies altered values from the wrong source. In a reference-list task, it cross-checks earlier corrections but misses an image-only withdrawal and adopts the canceled list. In an employee-profile task, an agent mistakes records repeating one source for independent confirmation. These failures, together with the need to support continued agent improvement, motivate three challenges.

• **Recovering cross-file context.** A document and the evidence that invalidates it can differ in location, format, and time. Summarizing each file separately leaves this relationship implicit. The challenge is to connect related records while retaining the source evidence needed to interpret and verify their relationship.

• **Learning reliable evidence judgments.** The same title, version label, or confirmation can identify valid evidence in one environment and mislead in another. Even faithful extraction can therefore reproduce an incorrect source. The challenge is to learn transferable reading and verification decisions, beyond surface matching or task-specific rules.

• **Sustaining useful challenges as agents improve.** As agents become more capable, a fixed set of environments may offer fewer opportunities to expose new failures. Yet modifying an environment does not necessarily produce a useful learning challenge: changes may remove essential evidence or invalidate the expected answer and evaluation checks. The challenge is to introduce new reasoning demands while keeping the task solvable and its outcome verifiable.

There are some existing works that improve agent learning with environment-relevant design. EnvHarness reshapes interactions while preserving underlying environment logic and verifiers (Huang et al., 2026); RLVE adapts generated problem difficulty, and EvoEnv synthesizes executable, validated environments (Zeng et al., 2026; Shi et al., 2026). Our focus is persistent file evidence: recovering its relationships, learning how to judge it, and changing its state together with the corresponding answers and checks.

Our Methodology. We present ENV-RETHINK, which prepares and evolves agent environments through three complementary capabilities. In environment preparation stage, ENV-RETHINK constructs a *Collection Map* through semantic regrouping and an *Event Log* through evidence-reviewed synthetic workflows. Additionally, ENV-RETHINK conducts post-training on qualified tool-using trajectories for verifying file authority and version relationships. In environment evolution stage, ENV-RETHINK creates plausible incorrect choices together with resolving evidence and updated evaluation assets. These capabilities support both reliable task execution and verifiable challenge generation.

We evaluate ENV-RETHINK on tasks adapted from Workspace-Bench and on Terminal-Bench 2.1 (Merrill et al., 2026). Across nine downstream models, our 27B preparation model raises mean rubric pass rate from 57.6% without preparation to 72.7%, exceeding unadapted-model preparation by 13.3 percentage points. Held-out file-verification accuracy rises from 56.2% to 76.5%. Environment evolution lowers success for at least three of four models on 32 of 55 retained tasks with paired results under unchanged requests.

2 PRELIMINARIES AND LIMITATION ANALYSIS

2.1 AGENT-NATIVE ENVIRONMENT

Component Distinctions. An agent system comprises a task model, a harness, and an environment. The task model M_θ performs reasoning and generates responses or tool-call requests; the harness H manages context, executes tool calls, and controls the execution budget. Together, they form the task agent A, whose output for a request q is $\hat{y} = A(q, E)$. *The environment comprises the persistent resources and state that the agent accesses and manipulates through the harness.* We represent it as $E = (F, S)$, where F contains files and service implementations and S denotes their current state.

Making an environment agent-native requires two complementary capabilities: *environment preparation*, which makes existing evidence easier to locate and interpret for reliable execution, and *environment evolution*, which creates new, verifiable challenges for continued learning. Both operate on the environment while preserving the task request, as illustrated in Figure 2.

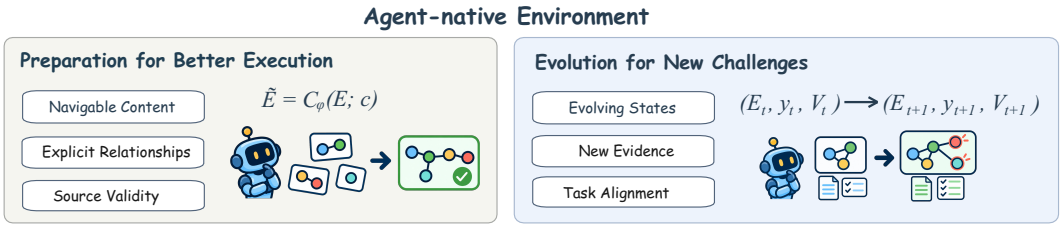

Figure 2: **Two complementary roles of an agent-native environment.** Preparation organizes existing evidence for task execution; evolution changes environmental states together with their reference outcomes and evaluators. Both preserve the task request.

• ***Environment preparation*** transforms an existing environment into $\widetilde{E} = C_\phi(E; c)$, where C_ϕ is a preparation operator and c denotes optional auxiliary information, such as metadata or task-related supervision. It supports three properties: *navigable content*, making relevant materials easy to locate; *explicit relationships*, exposing how records support, revise, or depend on one another; and *source validity*, providing inspectable evidence of authority and version applicability. Preparation supplies these properties through file selection, organization, and evidence-linked context, preserving the instance's correctness objective. The operator may use a trained preparation model without updating the downstream task model.

• ***Environment evolution*** changes environmental states to generate new, verifiable instances under the same request. It supports *evolving states*, through changes to files, records, or applicable conditions; *new evidence*, introducing further relationships to resolve; and *task alignment*, keeping reference outcomes and evaluation checks consistent with the changed state. We express this process as $(E_t, y_t, V_t) \rightarrow (E_{t+1}, y_{t+1}, V_{t+1})$, where y_t and V_t denote the reference outcome and evaluator. The task agent receives only the resulting environment and unchanged request; reference outcomes and evaluation assets remain separate for validation. These instances provide new challenges and verification targets for subsequent model improvement.

2.2 Limitations of Existing AI Agents

We evaluate nine model-harness configurations on 30 difficult tasks adapted from Workspace-Bench (Tang et al., 2026), comparing ground-truth-selected files (*Clean*) with all available files (*Noise*) under fixed execution settings. Figure 3 shows that all nine configurations perform worse in noisy environments, with mean rubric pass rate falling from 83.9% to 57.6%.

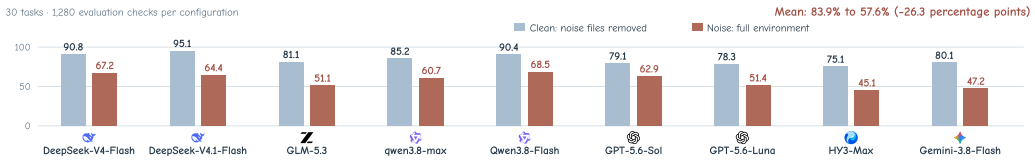

Figure 3: **Agent performance in clean vs. noisy environments.** Pass rates over 1,280 evaluation checks across 30 tasks for each of nine model-harness configurations. Clean uses ground-truth-selected files; Noise uses the full environment. Mean pass rate falls from 83.9% to 57.6%, a decrease of 26.3 percentage points.

Failure analysis reveals three recurring behaviors: (1) *Incomplete search and reading*: agents overlook corrections, constraints, and supporting evidence. (2) *Trusting filenames*: agents treat names such as `final.bib` as evidence of validity despite a later withdrawal notice. (3) *Unchecked reuse*: agents reuse similar files without verifying task requirements, carrying incompatible assumptions into their outputs.

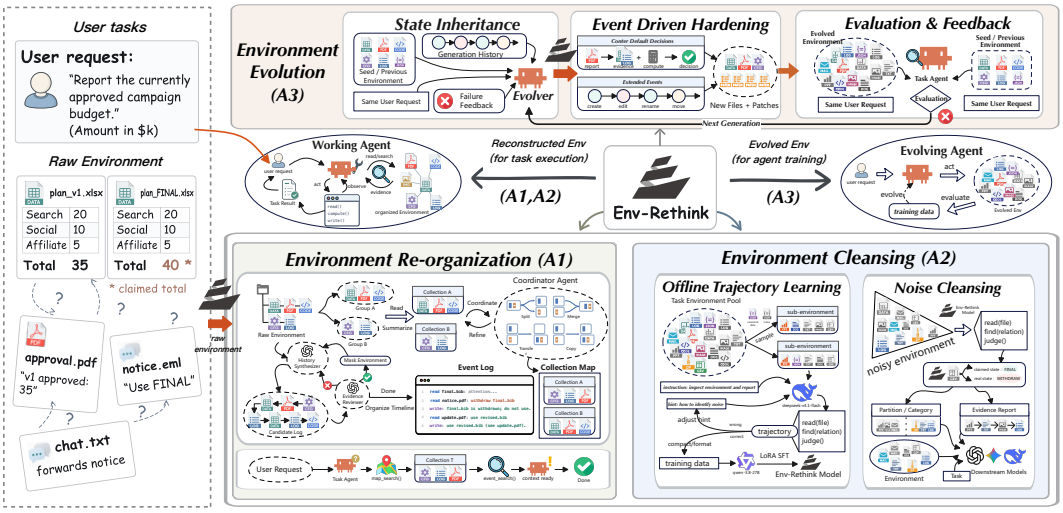

Figure 4: **Overview of ENV-RETHINK.** Environment preparation combines reorganization (A1), which constructs Collection Maps and Event Logs, with cleansing (A2), which learns file verification from offline teacher trajectories and supplies selected evidence and a report to a downstream task agent. Environment evolution (A3) extends generation histories through event-driven changes and evaluates the resulting environments under the same user request. The budget example illustrates conflicting evidence in a raw environment.

3 SYSTEM OVERVIEW

Figure 4 presents ENV-RETHINK with three complementary components: *environment reorganization* (A1) exposes cross-file context through Collection Maps and Event Logs; *environment cleansing* (A2) learns file verification from qualified teacher trajectories and supplies selected evidence with a preprocessing report; and *environment evolution* (A3) generates new states with updated reference outcomes and evaluation checks. These components support two agent workflows.

► **(1) Working Agent.** The agent uses the environment prepared through reorganization and cleansing to locate related records, inspect evidence, and execute the user request. Its tool interactions produce the final task result.

► **(2) Evolving Agent.** The agent executes the same request in evolved environments. Evaluation against updated reference outcomes and checks provides failure feedback for subsequent evolution, while validated environments supply new challenges for further agent training.

4 METHODOLOGY

ENV-RETHINK addresses two complementary problems: preparing reliable evidence for task execution and generating harder, verifiable environments for continued learning. The preparation module combines *reorganization*, which exposes cross-file context, with *cleaning*, which learns to verify and select evidence. The evolution module changes environmental states together with their reference outcomes and evaluation checks while preserving the user request.

4.1 ENVIRONMENT PREPARATION

Reorganization: exposing cross-file context. The reference-list case in Figure 1(b) requires both locating the relevant records and establishing that the withdrawal invalidates the list. Directory proximity or separate summaries need not reveal this relationship. We combine coverage-preserving semantic regrouping (*Collection Map*) with evidence-reviewed synthetic histories (*Event Log*), without changing source files.

Semantic regrouping. Deterministic grouping by directory and file count initializes collections. A *Collection Summarizer* reads their files and writes cards with themes, contents, representative evidence, and source paths. The *Semantic Coordinator* compares cards and member paths to propose splits, merges, transfers, or overlapping membership when supported by summaries or paths. Deterministic checks preserve every file in at least one collection. After each change, the *Collection Refiner* rereads affected groups and updates the cards for the next round. This feedback keeps summaries consistent with membership as groups change beyond directory boundaries.

Evidence-reviewed histories. The *History Synthesizer* reads files and proposes reading, writing, copying, and exporting events linked to source evidence. Workflows and timestamps are marked synthetic; quotations remain distinct from inferred relationships. Deterministic checks validate event structure and file references, while an independent *Evidence Reviewer* checks excerpts, actions, and relationships against the files. Feedback triggers repair; candidates exceeding the repair budget are discarded. The *Timeline Integrator* merges accepted histories while preserving causal order. Thus, in the reference-list case, a later synthetic timestamp alone cannot establish withdrawal: the relationship requires supporting source evidence.

Inputs and execution. Environment-Grounded Synthesis (EGS) uses only files and metadata. Task-Guided Synthesis (TGS) additionally uses clean/noise file labels for grouping and task requests and rubrics for history synthesis, simulating task-informed organization; instructions prohibit reproducing rubric entries in the public log. Both expose the same interface: the *Task Executor* browses cards, retrieves sessions by file path, checks original evidence, and executes the request with native tools. Tool-use hooks also surface this context during execution.

Cleaning: learning evidence-based judgments. Organization alone cannot establish whether a purported final version is valid or an authority claim is supported. We therefore post-train a file verifier to inspect supporting and conflicting records. Given a collection, manifest, and reading tools, it predicts each file's standard/noise partition, category, observed stage, family, timeline, and supporting evidence. Separating the status claimed by a file from its status relative to other records discourages reliance on filenames or apparent authority. Verification excludes the downstream request; the task agent determines task relevance.

Offline post-training. Independently sampled files may omit the evidence needed for verification. We instead construct sub-environments that retain authoritative records with their decoy chains, version families, and redirection evidence. Teachers inspect these collections through tools, producing trajectories of file access, observations, and judgments. Qualification checks verify reading fidelity, classification accuracy, retention of authoritative records, and version order. For failed trajectories, a separate agent uses annotations to revise the teacher's hint; unsuccessful retries are discarded. Student prompts omit these teacher hints.

We apply LoRA (Hu et al., 2022) fine-tuning to Qwen3.8-27B using the qualified trajectories:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{\sum_{t=1}^{T} m_t \log p_\theta(x_t \mid x_{<t})}{\sum_{t=1}^{T} m_t}, \tag{1}$$

where m_t selects assistant tokens, including tool calls and judgments; tool observations provide context without contributing to the loss. At deployment, verification produces selected files and a preprocessing report documenting the evidence and exclusions. A separate downstream agent executes the original request using this prepared environment.

4.2 Environment Evolution

Creating answerable challenges. Changing files can introduce ambiguity or invalidate existing evaluation checks. We organize evolution around *counter-default decision points*: each pairs a plausible incorrect choice with evidence that uniquely resolves it and a check that rejects it. These points concern effective dates, exclusions, exception rules, or numerical recalculation. Generation instructions require at least three points per variant and evidence chains no shorter than those in the seed.

Joint state and reference updates. The evolver uses the current environment, generation history, and available failure feedback to produce file changes, updated reference outcomes, and revised

checks under the unchanged request:

$$(E_t, y_t, V_t) \rightarrow (E_{t+1}, y_{t+1}, V_{t+1}).$$

File operations are recorded as events, allowing successive variants to build on earlier changes. A deterministic merger checks history continuity before assembling the new environment. This generation history, reference outcomes, and evaluation assets remain separate from the task agent's inputs.

Validation and feedback. Structural checks verify assembly, references, evaluator syntax, and the unchanged request. Answer validation requires the updated reference solution to pass and the previous expected output to fail; shortcut audits check for leaked conclusions. Matched seed–variant runs then measure difficulty under identical agent configurations and budgets. Failed-check summaries guide subsequent evolution, while validated instances provide executable challenges and verification targets for further learning.

5 EXPERIMENTS

5.1 EVALUATION OVERVIEW

We evaluate environment organization, learned environment preparation, and environment evolution in separate studies. The motivating Clean/Noise comparison appears in Section 2.2. The following subsections retain each study's inputs, evaluation conditions, and reported results; detailed training and downstream evaluation settings appear in Appendix B, and the evolution protocol appears in Appendix C.

We refer to the 30-task set in Section 2.2 as Environment-Hard. The Clean and Noise settings are the GT and Full conditions in Appendix B.6. Figure 3 shows that every configuration scores lower in Noise: rates range from 45.1% to 68.5%, compared with 75.1% to 95.1% in Clean. The mean difference is 26.3 percentage points.

5.2 ENVIRONMENT ORGANIZATION

5.2.1 COMPARING THE TWO SYNTHESIS MODES

Table 1 compares Task-Guided Synthesis (TGS), Environment-Grounded Synthesis (EGS), and the original noisy environment without the added organization layer (Noise). Both synthesis modes provide Collection Map, Event Log, and tool-use hooks that surface environment context during execution; TGS additionally uses task-specific supervision during preparation.

5.2.2 SYNTHESIS RESULTS

The evaluation covers 30 tasks with 1,280 rubric items for every model. TGS outperforms EGS for all five models. Both modes exceed the Noise baseline for all five models. DeepSeek-V4.1-Flash improves from 64.4% in Noise to 82.9% with EGS and 87.3% with TGS; GLM-5.3-Flash improves from 58.3% to 63.6% and 81.6%, respectively. For GPT-5.6-Sol, the complete 30-task comparison gives 78.2% for TGS versus 69.6% for EGS, a difference of 8.6 percentage points. Component ablations appear in Appendix B.7.

5.2.3 FROM CONTEXT SYNTHESIS TO ENVIRONMENT-UNDERSTANDING TRAINING

The two modes identify the next research question: how can a preparation model recognize valid evidence from the environment's source files and reliably generate useful context? Task-guided synthesis outperforms EGS in the matched comparisons, indicating that the additional task-specific supervision improves the utility of the prepared context. This motivates learning to identify relevant evidence and resolve file relationships from materials in the environment.

Module 2 therefore turns these environmental judgments into training targets: acquiring evidence, identifying source authority, resolving version relationships, and preparing environments through file selection and structured reports, making environment preparation learnable and reusable.

Table 1: TGS, EGS, and Noise rubric pass rates (%), with passed/total checks in parentheses. TGS and EGS use the full environment layer (Collection Map, Event Log, and tool-use hooks). All reported scores cover the same 30 tasks and 1,280 rubric items. **Bold** denotes the best result in each column, and underlining denotes the second best. Δ reports the percentage-point difference between TGS and Noise.

Model	TGS	EGS	Noise	Δ
Harness: Claude Code				
DeepSeek-V4.1-Flash	**87.3** (1117/1280)	**82.9** (1061/1280)	**64.4** (824/1280)	**+22.9**
GLM-5.3-Flash	81.6 (1044/1280)	63.6 (814/1280)	58.3 (746/1280)	**+23.3**
Claude Opus 4.8	79.6 (1019/1280)	71.3 (913/1280)	60.5 (774/1280)	**+19.1**
HY3	62.0 (793/1280)	46.3 (593/1280)	45.1 (577/1280)	**+16.9**
Harness: Codex				
GPT-5.6-Sol	78.2 (1001/1280)	69.6 (891/1280)	62.9 (805/1280)	**+15.3**

5.3 Learned Environment Preparation

We evaluate Qwen3.8-27B and the Env-Rethink model on training-source and held-out task environments, then measure downstream task performance using the prepared environments.

5.3.1 Module Performance

We evaluate Qwen3.8-27B and the Env-Rethink model on the complete file collections of environments from 15 training-source tasks (1,579 files) and 15 held-out tasks (617 files). On training-source environments, partition accuracy increases from 74.0% to 90.0%. On held-out environments, it increases from 56.2% to 76.5%, and joint partition, category, and stage accuracy rises from 19.4% to 32.4%. Table 2 summarizes the comparison; Appendix B.4 describes the task split and scoring protocol.

Table 2: File-verification accuracy (%) on training-source and held-out environments. Each split contains 15 tasks, with 1,579 and 617 files, respectively. **Bold** denotes Env-Rethink. Δ reports the gain over Qwen3.8-27B in percentage points (pp).

	Training-source environments			Held-out environments		
Metric	Qwen3.8-27B	Env-Rethink	Δ	Qwen3.8-27B	Env-Rethink	Δ
Partition accuracy	74.0	**90.0**	**+16.0**	56.2	**76.5**	**+20.3**
Noise-category accuracy	39.8	**56.8**	**+17.0**	33.4	**38.1**	**+4.7**
Observed-stage accuracy	68.2	**72.0**	**+3.8**	26.1	**59.7**	**+33.6**
Joint accuracy	27.5	**45.5**	**+18.0**	19.4	**32.4**	**+13.0**

5.3.2 Downstream Utility

We evaluate downstream utility on the 30-task Environment-Hard set with nine downstream models from the DeepSeek, GLM, Qwen, GPT, HY, and Gemini families. We compare three environment conditions: *Full environment* retains all noisy source files; *Qwen3.8-27B* uses the unadapted model to select files; and Env-Rethink uses the trained model for file selection and evidence reporting. Each model is evaluated on the same 30 tasks under all three conditions.

Rubric pass rate pools passed checks over the 1,280 checks across these tasks. Figure 5 presents the comparison, with per-model counts and aggregation details in Appendix B.6.

Across the nine models, the mean pass rate rises from 57.6% with full environments and 59.4% with Qwen3.8-27B to 72.7% with Env-Rethink. Env-Rethink improves over Qwen3.8-27B by 13.3 percentage points on average, with gains across all nine models. Improvements over full environments range from 8.4 to 18.5 percentage points. These results show the downstream utility of the environments prepared by Env-Rethink through file selection and evidence reporting.

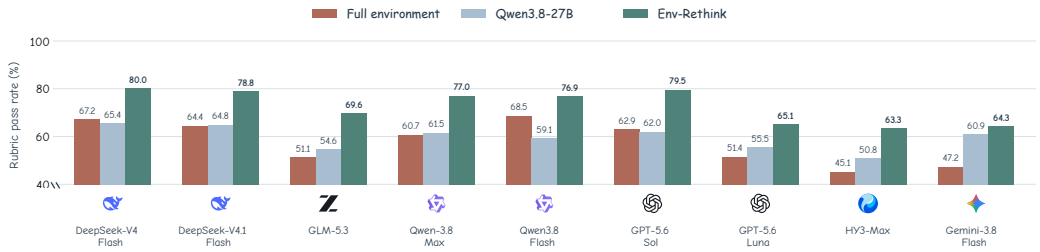

Figure 5: Downstream rubric pass rates for nine models on the same 30 tasks. The legend identifies the environment preparation condition; the horizontal axis lists downstream task models. Each condition includes 1,280 checks per downstream model. The vertical axis omits 0–40%, as indicated by the axis break.

5.3.3 Transfer and Reuse

The same Env-Rethink model prepares environments for all nine downstream models across the 30-task evaluation set. Its reuse occurs at the environment layer: file judgments made during preparation determine the evidence available to each task agent. The consistent improvements across downstream models show the utility of this shared preparation step.

Module 2 produces a learned component that can be evaluated and versioned within an environment. Environment evolution uses virtual event histories to evolve seed states into harder instances under the same task request, creating further examples for environment understanding and downstream work.

5.4 Environment Evolution

We apply event-driven evolution to 59 Terminal-Bench 2.1 (Merrill et al., 2026) seed tasks spanning 14 categories, including software engineering, debugging, model training, and data processing. The evolved environments extend generation histories, materialize changed states, and recompute reference outcomes while retaining the task requests. We compare seed and evolved environments using DeepSeek-V4.1-Flash, Gemini-3.8-Flash, Qwen3.8-Flash, and GLM-5.3-Flash, with a nominal three trials per arm and native binary task-success scoring. The analysis excludes a task with a failing variant reference run, leaving 55 tasks with paired results. Appendix C.1 gives the aggregation rules, task coverage, and validation scope.

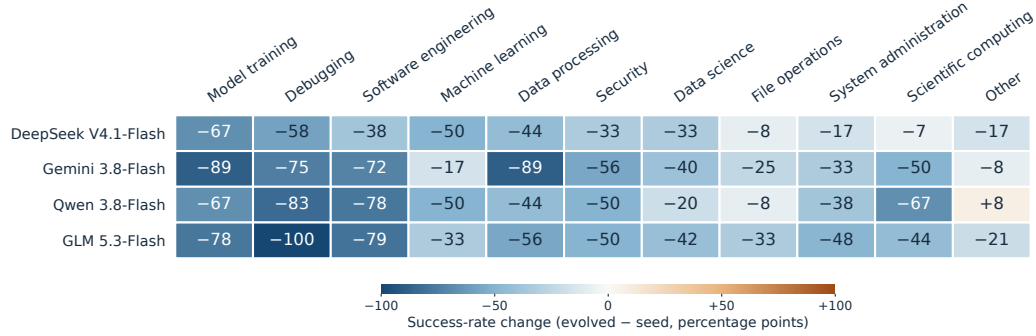

Figure 6: **Environment evolution on Terminal-Bench 2.1.** Cell values give the reported mean change in success rate (evolved minus seed, percentage points), with models in rows and task categories in columns. Blue indicates lower success after evolution; orange indicates higher success. Means use available pairs with nonzero seed success. Other combines games, personal assistant, mathematics, and optimization. Evaluation settings and validation details appear in Appendix C.1.

Harder environments across tasks and models. Of the 55 retained tasks with paired results, 32 (58.2%) show lower success in at least three models, including 21 with decreases in all four models. Four further tasks show decreases in exactly two models. These effects span ten categories: 10 of 13 software-engineering tasks, four of five debugging tasks, and all three model-training tasks meet the three-model criterion (Figure 6). The category-level changes show that evolution can expose new execution failures across distinct technical domains under the same task requests.

Environment organization and learned preparation help agents recover valid evidence; environment evolution creates new situations in which that ability is tested. The preparation results also reveal a useful balance between evidence retention and noise filtering (Table 6). Across the 30 environments, ENV-RETHINK retains 160 of 192 standard files, compared with 136 for Qwen3.8-27B, while reducing admitted noise files from 606 to 271. The resulting workspaces contain 431 files rather than 742, giving downstream agents a smaller collection to inspect while preserving more standard evidence. Together, these studies support environment preparation and challenge generation as resources for continued improvement. Execution failures can inform subsequent generation, and regenerated reference outcomes supply verification targets. This provides the challenge-generation component of environment-driven recursive self-improvement.

6 RELATED WORK

Agent tools and context management. Agent interfaces determine what information can be found and carried across sessions. OpenAI provides file search, computer use, orchestration, and execution tracing (OpenAI, 2025). Anthropic uses selective retrieval, persistent notes, and structured handoffs (Rajasekaran et al., 2025; Young, 2025; Rajasekaran, 2026), while SWE-agent designs interfaces for repository navigation and editing (Yang et al., 2024). Together, these systems make information access and continuity part of agent execution.

Environment adaptation and generation. Constructed environments define the interactions available to agents and the outcomes that can be checked. EnvHarness adapts existing environments while preserving their logic and verifiers (Huang et al., 2026). RLVE and AgentScaler generate training tasks and tool-use simulations (Zeng et al., 2026; Fang et al., 2026), while AWM and EvoEnv produce stateful or executable environments (Wang et al., 2026; Shi et al., 2026). Across these approaches, environment structure and verifiability are explicit design choices.

Learning from interaction. Interaction records can turn observations and corrections into reusable guidance. ReAct grounds decisions in tool observations, and Reflexion retains textual feedback for later trials (Yao et al., 2023; Shinn et al., 2023). OpenAI and Thrive Holdings' Tax AI turns practitioner corrections into evaluated fixes (Srinivasan et al., 2026); Factory's Signals identifies recurring failures across agent sessions and proposes code changes (Factory Research, 2026). The value of a trace depends on how it informs later decisions.

Feedback-guided environments. Agent performance can guide which challenges an environment presents next. POET generates challenges alongside solution optimization and transfers solutions across environments (Wang et al., 2019). PAIRED uses differences in agent returns to generate solvable challenges (Dennis et al., 2020). Both link environment difficulty to observed learner behavior.

7 CONCLUSION

In this paper we studied how fragmented and unreliable evidence limits LLM agents in persistent file-based environments. We proposed Env-Rethink to prepare these environments for reliable execution and evolve them into harder, verifiable challenges. Env-Rethink supported (1) Collection Maps and evidence-reviewed Event Logs for organizing cross-file context, (2) a 27B model post-trained on tool-use trajectories for verifying and selecting evidence, and (3) event-driven evolution that updates environmental states, reference answers, and evaluation checks under unchanged requests. Empirically, learned preparation improved held-out file-partition accuracy from 56.2% to 76.5% and mean rubric pass rate from 57.6% to 72.7% across nine models on 30 tasks. Evolution reduced success for at least three of four models on 32 of 55 paired Terminal-Bench tasks. These results support environment preparation and evolution as complementary foundations for more reliable agents and future recursive self-improvement.

AI USE STATEMENT

We used AI tools in three ways. First, language models are core components of ENV-RETHINK for environment organization, file verification, and event-driven evolution; we fine-tune Qwen3.8-27B as the environment-preparation model. Second, models from the DeepSeek, GLM, and Gemini families generate teacher trajectories, and AI tools assist with file annotation and synthetic environment construction. These outputs undergo the qualification, consistency checks, and human annotation review described in the methodology and appendices. Third, AI tools assist with manuscript drafting and editing, figure preparation, and literature and reference checks. The authors take responsibility for the final content of this paper, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies agent reliability in environments that simulate the complexity of real-world workspaces, including fragmented information, conflicting versions, and outdated records. Experiments use tasks adapted from Workspace-Bench and Terminal-Bench. Constructed workflows and event histories are identified as simulated and linked to source evidence. Data annotation includes consistency checks and human review of challenging cases and disagreements. These procedures support traceable environment construction and evaluation of agents' ability to identify and use reliable evidence.

REPRODUCIBILITY STATEMENT

Our anonymous repository is available at `https://anonymous.4open.science/r/Env-Rethink-EDAD/`. The methodology describes environment organization, trajectory qualification, model adaptation, and event-driven evolution. Appendix A documents data selection, annotation definitions, and sub-environment construction. Appendix B provides training hyperparameters, hardware, trajectory preparation, evaluation splits, scoring rules, and per-model downstream results. Appendix C specifies the evolution protocol, task coverage, and aggregation rules. Together, these details describe how the reported environments, training corpus, and evaluation results are constructed.

REFERENCES

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart J. Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 33, pp. 13049–13061. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html`.

Factory Research. Signals: Toward a self-improving agent. Factory, January 2026. URL `https://factory.com/news/factory-signals`.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Ningyu Zhang, Fei Huang, Wentao Zhang, and Jingren Zhou. Towards general agentic intelligence via environment scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17610–17621, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.872. URL `https://aclanthology.org/2026.findings-acl.872/`.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

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. *arXiv preprint arXiv:2608.19880*, 2026. URL https://arxiv.org/abs/2608.19880.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Moonshot AI. Kimi K3: Open frontier intelligence. Technical blog, 2026. URL https://www.kimi.com/en/blog/kimi-k3.

OpenAI. New tools for building agents. Product and engineering announcement, March 2025. URL https://openai.com/index/new-tools-for-building-agents/.

Prithvi Rajasekaran. Harness design for long-running application development. Anthropic Engineering, March 2026. URL https://www.anthropic.com/engineering/harness-design-long-running-apps.

Prithvi Rajasekaran, Ethan Dixon, Carly Ryan, and Jeremy Hadfield. Effective context engineering for AI agents. Engineering report, September 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Yucheng Shi, Zhenwen Liang, Kishan Panaganti, Dian Yu, Wenhao Yu, and Haitao Mi. Learning to build the environment: Self-evolving reasoning RL via verifiable environment synthesis. *arXiv preprint arXiv:2605.14392*, 2026. URL https://arxiv.org/abs/2605.14392.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Aravind Srinivasan, Samay Shamdasani, Arthur Fernandes Araujo, and John de Wasseige. Building self-improving tax agents with codex. OpenAI, May 2026. URL https://openai.com/index/building-self-improving-tax-agents-with-codex/.

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. *arXiv preprint arXiv:2605.03596*, 2026. doi: 10.48550/arXiv.2605.03596. URL https://arxiv.org/abs/2605.03596. Preprint.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. POET: Open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–151, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3321707.3321799. URL https://doi.org/10.1145/3321707.3321799.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.10090.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent–computer interfaces enable automated software engineering. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Justin Young. Effective harnesses for long-running agents. Engineering report, November 2025. URL https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, Chenyang Zhao, Yulia Tsvetkov, Simon Shaolei Du, Natasha Jaques, Hao Peng, Pang Wei Koh, and Hannaneh Hajishirzi. RLVE: Scaling up reinforcement learning for language models with adaptive verifiable environments. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2511.07317.

A DATA

This appendix specifies the annotation and collection procedures supporting the file-verification module.

A.1 LABEL DEFINITIONS

Each annotation records partition, category, observed stage, family, and evidence. The six categories describe a file's authority and relationships within the supplied collection.

Table 3: File-verification category definitions.

Category	**Meaning**
authoritative record	The designated authoritative record, supported by source records, content consistency, or archive lineage.
false final version	A purported final or newest version contradicted by the genuine version chain or its own contents.
fabricated authority	Fabricated approvals, messages, or other authority claims lacking supporting records.
misleading redirection	A pointer or instruction conflicting with annotated authority relationships or the timeline.
superseded version	A replaced member of a version family that may still supply historical evidence.
out-of-scope file	A normal file outside the collection's target scope; task relevance is assessed for the specific downstream task.

A.2 DATA MANIFEST

The construction rule allows identical file sets at most twice; file, family, and semantic overlap are checked across sub-environments. Repeated teacher trajectories are handled separately through the retention procedure in Appendix B.3.

A.3 SELECTION

Training selection starts from the 30-task, 1,280-rubric Environment-Hard pool and favors strong clean performance with substantial noise sensitivity. The record contains 12 core tasks with 1,235 files and four reserve tasks, totaling 16 tasks and 1,597 files. Task 161 is excluded from the 15-task construction pool, which contains 1,579 files. The original 209 sub-environments and 30 subsequently generated held-out environments form a collection of 239. Qualified teacher trajectories cover 139 environments; the 30 additional environments and the 70-environment teacher-failure pool are excluded from SFT training. The full-environment module evaluation uses the remaining 15 tasks, with 617 labeled files, from the 30-task set. These parent tasks are separate from the 15-task training pool.

Selection uses Clean $\geq$90%, a reported Noise$-$Clean contrast of -38 to -75 percentage points, and file-level evidence of the failure mechanism.

A.4 ANNOTATION

Filename and path rules initialize stage and category; task-analysis reports supply strong-decoy evidence; teacher labeling covers remaining files. The process then checks manifests and family constraints. Strong decoys and disagreements receive human review, while other files receive category-stratified checks. The construction record reports that all 1,597 labeled files passed consistency checks.

A.5 SUB-ENVIRONMENTS

Attack scenes retain authoritative targets with decoys and fabricated supporting chains. Version families retain related authoritative and superseded records with their order. Redirect groups retain pointers, targets, and supporting messages. Fillers add density around these relationships. Composition varies the number of authoritative records and checks whether visible evidence supports each label.

Small batches contain 4–12 files, medium batches 8–30, and large batches at most 80. The specified mixture is 45%/35%/20%. Size, semantic composition, and prompt condition accompany each generation and acceptance record.

A.6 TEACHER READING

Current generation uses DeepSeek, GLM, and Gemini teacher families through Claude Code in isolated sandboxes. Complete messages retain calls, observations, and final judgments. Read-fidelity checks connect each file judgment to content-access records and supporting observations.

The generation specification records three hint levels: L0 supplies the verification task, L1 adds category features and distinguishes claimed stage from verified category, and L2 supplies a checking sequence. The current student training prompt contains the verification instruction, schema, and reading requirements without discriminative guidance. Prompt provenance records which generation conditions produced each accepted trajectory and how it entered the final training corpus.

For retries, a separate agent examines the failed teacher trajectory and the ground-truth annotations to revise the hint. The teacher receives the revised hint and repeats the verification task.

A.7 QUALIFICATION AND OUTPUTS

After read fidelity passes, deterministic qualification requires partition F1 $\geq$0.9, zero false rejections of authoritative records, correct classification of all strong decoys, and correct order where a version family occurs. The strong-decoy categories are *false final version*, *fabricated authority*, and *misleading redirection*. The generation specification permits at most two additional retries with hints revised by the separate agent described in Appendix A.6; remaining failures are retained for analysis.

The current record reports 2,116 qualified trajectories. The dataset links their messages and qualification records to file, family, scene, annotation, and sub-environment manifests.

A.8 UNSEEN-TASK EXTENSION

The extension contains tasks 72, 75, 78, 85, 100, 146, 159, 171, and 266, none of which entered training, and records annotations for 304 files. Six existing-noise tasks (85, 100, 146, 159, 171, and 266) contribute 224 files: 70 standard files and 154 noise files across 42 version families. The other three tasks (72, 75, and 78) receive 68 generated noise files across 18 version families. The construction record includes solvability and task-navigation audits for these tasks.

The extension tasks are included in the unified 30-task downstream evaluation in Appendix B.6.

B REPRODUCIBILITY

B.1 TRAINING CONFIGURATION

The training run uses model-native tool-call serialization: tool calls are encoded as text in assistant content so that the rendered training messages preserve them. The ENV-RETHINK model is obtained by adapting Qwen3.8-27B with LoRA.

Table 4: Configuration of the completed supervised fine-tuning run.

Item	Current configuration
Base model	Qwen3.8-27B; 27B dense model; recorded context capacity 256k
Adaptation	LoRA supervised fine-tuning on multi-turn tool trajectories
Epochs	3
LoRA rank / alpha	64 / 128
Target modules	All linear layers
Training examples / development examples	1,780 / 201
Learning rate	1×10^{-4}, cosine schedule
Numerical precision	bf16
Training context	32k
Hardware	One machine with 8 H20 GPUs; DeepSpeed ZeRO-3
Optimization steps	168
Training time	9 h 44 min 46 s

B.2 Optimization Diagnostics

The run completed three epochs in 168 optimization steps. The final training loss was 0.199 and training token accuracy was 93.6%; the final development loss was 0.197 and development token accuracy was 94.1%. Training curves cover 168 steps, with development evaluation at optimization steps 30, 60, 90, 120, 150, and 168.

The final evaluation is reported below. Figure 7 shows the loss and token-accuracy curves.

Table 5: Final optimization and development metrics.

Run	Epochs	Train loss	Train token acc.	Dev. loss	Dev. token acc.
Final	3	0.199	93.6%	0.197	94.1%

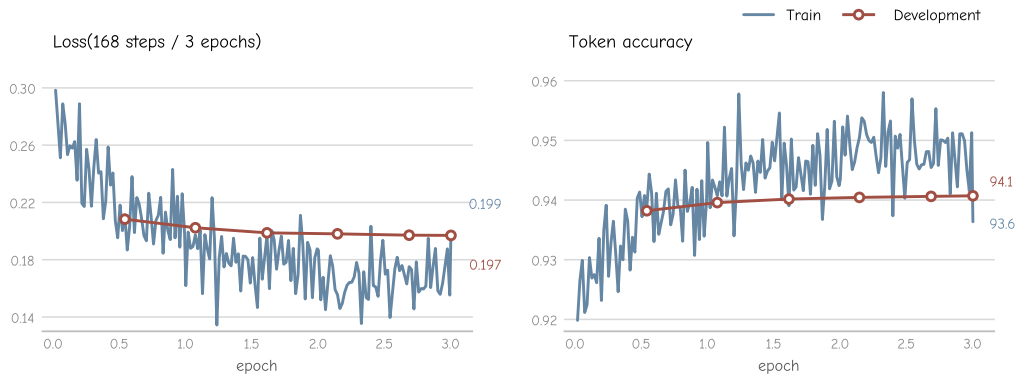

Figure 7: Training and development loss and token accuracy over 168 steps (3 epochs) of the fine-tuning run. Development evaluation occurs at optimization steps 30, 60, 90, 120, 150, and 168. Final training loss/token accuracy are 0.199/93.6%, and final development values are 0.197/94.1%.

B.3 Trajectory Preparation and Deployment

Trajectory cleaning truncates only excessively long tool outputs while preserving each invocation/response pair. Examples that remain overlength are discarded as complete samples; training applies no further truncation. This preparation retains complete conversational structure within the

training context. When a sub-environment has several teacher trajectories, a small subset is retained by rotating among teachers, controlling repeated exposure to the same environment while retaining variation in reading and reasoning.

The assembled corpus contains 1,780 training and 201 development examples. The reported deployment merges the LoRA adapter into the base model after optimization step 168. Module evaluation below reports results on full task environments.

B.4 Module Evaluation Details

Table 2 evaluates the complete file collections of environments from both the 15 training-source parent tasks and the 15 held-out parent tasks, using the updated file annotations. The 30-task annotation set contains 2,196 files: 1,579 from the 15 training-pool tasks and 617 from the held-out tasks. We pool correct predictions over eligible files within each split. For training-source environments, partition and joint accuracy use all 1,579 files, noise-category accuracy uses 1,492 noise files, and observed-stage accuracy uses 1,328 files with a nonempty stage label. For held-out environments, partition and joint accuracy use all 617 files, noise-category accuracy uses 512 noise files, and observed-stage accuracy uses 372 files with a nonempty stage label. Missing predictions count as incorrect. Joint accuracy requires the partition, category, and stage judgments to agree with the reference labels.

On training-source environments, the Qwen3.8-27B and ENV-RETHINK correct counts are respectively 1,169 and 1,421 for partition, 594 and 848 for noise category, 906 and 956 for stage, and 434 and 719 for joint accuracy. On held-out environments, they are 347 and 472 for partition, 171 and 195 for noise category, 97 and 222 for stage, and 120 and 200 for joint accuracy. All 15 held-out tasks contribute to these totals.

B.5 Environment Preparation Details

The preparation model selects files and supplies an evidence report for downstream use. We evaluate selection quality on 30 task environments containing 2,196 files: 192 standard files and 2,004 noise files. Table 6 reports retention and filtering quality. ENV-RETHINK retains 160 of 192 standard files and admits 271 noise files, compared with 136 and 606, respectively, for Qwen3.8-27B.

Table 6: Environment-preparation quality over 30 tasks and 2,196 files. Rejected standard files and admitted noise files are counts; recall is measured over standard-labeled files.

Preparation model	Selected	Standard rejected	Noise admitted	Standard recall
ENV-RETHINK	431	32	271	83.3%
Qwen3.8-27B	742	56	606	70.8%
Ground truth	192	0	0	100.0%

B.6 Thirty-Task Downstream Evaluation

The evaluation covers nine downstream models on the same 30 tasks. Full retains all files in the original environment; Qwen3.8-27B uses files selected by the unadapted model; ENV-RETHINK uses files and an evidence report prepared by our trained model. The model names identify environment preparation conditions. We also report a GT-curated environment reference, which selects files using ground-truth labels.

Per-model rates pool passed checks across tasks. Missing outputs count as zero passed checks while retaining the task's rubric denominator. Each condition therefore covers 11,520 checks across the nine models. Means are computed from unrounded counts. The reported scores incorporate corrected evaluator judgments. GT-curated scores serve as an empirical reference for each downstream model.

The task set is 72, 75, 78, 79, 85, 87, 94, 100, 108, 124, 129, 146, 154, 159, 160, 161, 171, 207, 258, 266, 267, 288, 291, 300, 314, 334, 357, 359, 372, and 374. The nine-task annotation extension in Appendix A.8 contributes to this evaluation set.

Table 7: Thirty-task downstream results. Full, Qwen3.8-27B, and ENV-RETHINK report passed checks out of 1,280 and the corresponding rate. GT gives the reference pass rate on environments prepared using label-based file selection. **Bold** denotes ENV-RETHINK, and underlining denotes the better of the two baselines (Full and Qwen3.8-27B). Δ is the difference between ENV-RETHINK and Qwen3.8-27B, in percentage points.

	Baselines			Reference	
Downstream model	Full	Qwen3.8-27B	**ENV-RETHINK**	GT	Δ
DeepSeek-V4-Flash	860 (67.2%)	837 (65.4%)	**1,024 (80.0%)**	90.8%	**+14.6**
DeepSeek-V4.1-Flash	824 (64.4%)	829 (64.8%)	**1,008 (78.8%)**	95.1%	**+14.0**
GLM-5.3	654 (51.1%)	699 (54.6%)	**891 (69.6%)**	81.1%	**+15.0**
Qwen-3.8-Max	777 (60.7%)	787 (61.5%)	**986 (77.0%)**	85.2%	**+15.5**
Qwen3.8-Flash	877 (68.5%)	756 (59.1%)	**984 (76.9%)**	90.4%	**+17.8**
GPT-5.6-Sol	805 (62.9%)	793 (62.0%)	**1,018 (79.5%)**	79.1%	**+17.6**
GPT-5.6-Luna	658 (51.4%)	710 (55.5%)	**833 (65.1%)**	78.3%	**+9.6**
HY3-Max	577 (45.1%)	650 (50.8%)	**810 (63.3%)**	75.1%	**+12.5**
Gemini-3.8-Flash	604 (47.2%)	780 (60.9%)	**823 (64.3%)**	80.1%	**+3.4**
Mean rate	57.6%	59.4%	**72.7%**	–	**+13.3**

B.7 TGS COMPONENT ABLATIONS

On the same 30-task set, removing Collection Map reduces DeepSeek-V4.1-Flash's rubric pass rate from 87.3% to 75.1% (-12.2 percentage points), while removing Event Log reduces it to 84.8% (-2.4 percentage points). Both ablations retain the tool-use hooks and the remaining environment component.

Table 8: DeepSeek-V4.1-Flash TGS component ablations on 30 tasks with 1,280 rubric items. All conditions retain tool-use hooks. Changes relative to the full condition are in percentage points, calculated from unrounded counts.

Condition	Passed/total	Rate (%)	Δ vs. full
Full (Collection Map + Event Log + hooks)	1117/1280	87.3	–
w/o Collection Map	961/1280	75.1	-12.2
w/o Event Log	1086/1280	84.8	-2.4

C EVENT-DRIVEN GENERATION PROTOCOL

The materialized environment and its generation record are represented separately:

$$E_t = \langle F_t, S_t \rangle, \qquad R_t = \langle L_t, y_t, V_t; \nu_t \rangle, \tag{2}$$

where F_t comprises files and service implementations, S_t is their state, L_t is the generation history, y_t is the reference outcome, V_t is the evaluator, and ν_t identifies the variant. The task agent receives E_t and the unchanged request q; generation and verification retain R_t.

Generation and validation. The Environment Evolver receives the seed or previous variant, its generation history, and available execution feedback. It produces environmental changes, causally linked events, and revised reference outcomes. Assembly applies the changes to produce a runnable environment. Structural checks verify event links, file references, evaluation assets, and byte-for-byte preservation of the task request. Reference execution checks that the new answer passes and the previous answer fails. Each variant specifies at least three counter-default decision points, linking a plausible incorrect choice to resolving evidence and an executable check. Shortcut audits inspect generated materials for conclusions that would bypass the seed's reasoning requirements. Evaluation uses the assembled environment, preserving its state when files are supplied to the agent.

C.1 Evaluation Settings

Task coverage. The collection contains 59 Terminal-Bench 2.1 seeds: four easy and 55 medium tasks. We exclude `sanitize-git-repo` because its variant reference run fails, leaving 58 seeds, of which 55 have paired results. Among these, 32 show a success-rate decrease of at least 20 percentage points in three or more models, four in exactly two models, and 19 in at most one model. The reported 58.2% uses the 55 retained paired tasks as its denominator. Category summaries apply the same exclusion. The aggregate reports observed success-rate changes. A seed-environment fault is recorded for `mteb-retrieve`.

Scoring and aggregation. We evaluate DeepSeek-V4.1-Flash, Gemini-3.8-Flash, Qwen3.8-Flash, and GLM-5.3-Flash with matched configurations and execution budgets across seed and variant. The target allocation is three independent trials per arm. Task success requires passing all native checks; check-level pass rates are used only for diagnosis. For task i and model m,

$$\Delta_{i,m} = 100\big(\hat{p}_{i,m}^{\text{evolved}} - \hat{p}_{i,m}^{\text{seed}}\big), \qquad h_i = \sum_m \mathbf{1}\{\Delta_{i,m} \le -20\}. \tag{3}$$

The main result counts tasks with $h_i \ge 3$. This is a screening criterion: one changed outcome in three trials corresponds to 33.3 percentage points. Rates use completed-trial denominators; for example, GLM's seed score on `largest-eigenval` is 1/2 and its variant score is 0/3.

Category means in Figure 6 average task-level contrasts, excluding missing pairs and pairs with zero seed success. Other combines games, personal assistant, mathematics, and optimization.