# BEYOND SKILL EVOLUTION: SELF-EVOLVING CONTEXT MANAGEMENT POLICIES FOR LONG-HORIZON AGENT HARNESSES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Harness evolution improves LLM agents by learning from execution trajectories, but existing experience- and skill-based methods are less effective on long-horizon tasks. As interactions grow, useful evidence can be buried by redundant or outdated context, making context management itself a key bottleneck. We introduce CONTEXTEVO, a framework that learns a context policy from long-horizon trajectories. CONTEXTEVO reconstructs the model-visible context at key decision points, identifies context-related failures, and applies targeted policy updates. Starting from the open-source Pi-agent harness, CONTEXTEVO improves performance across three long-horizon task benchmarks, achieving results comparable to or better than several prominent agent harnesses, including Codex, OpenCode, and OpenClaw. Additional analyses show that fixed or locally evolved context strategies can fall short under long-horizon information pressure, while our methods adapt to the information demands of each environment.

## 1 INTRODUCTION

LLM agents are increasingly capable of solving complex, long-horizon tasks, including web navigation, software engineering, and computer use (Zhou et al., 2024b; Jimenez et al., 2024; Yuan et al., 2026). In these tasks, strong agent performance depends not only on a capable base model, but also on a well-designed harness that manages context, tool use, and execution (Lin et al., 2026a; Lee

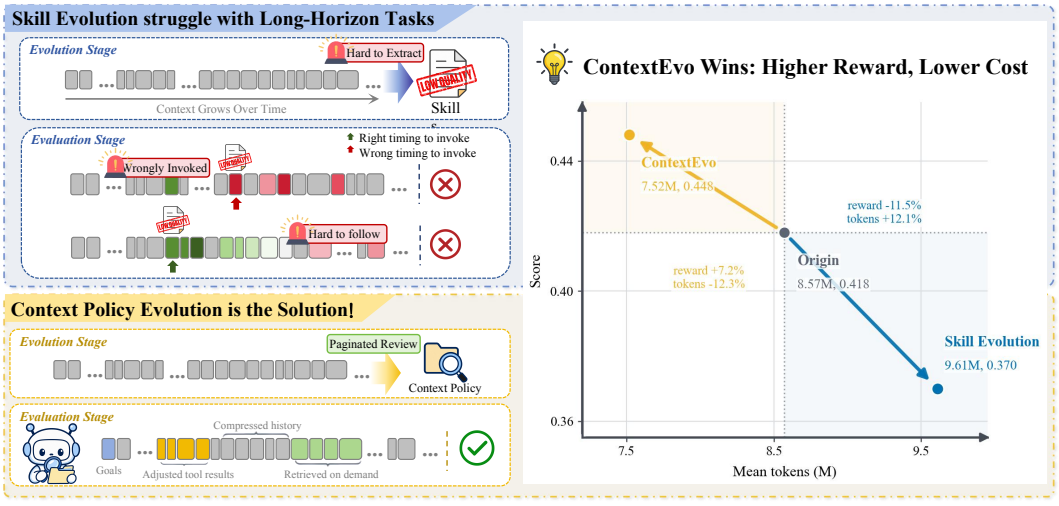



Figure 1: **From skill evolution to context-policy evolution.** As interaction histories grow, ***skill evolution*** may fail at both stages: extracting low-quality skills during evolution and invoking them at the wrong time or following them unreliably during evaluation. In contrast, our **CONTEXTEVO** evolves an efficient context policy, achieving higher scores with fewer tokens.



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.

et al., 2026). As static harnesses may struggle to adapt to different task requirements, recent work has explored harness evolution, where agent trajectories are used as feedback signals to improve harness components (Lee et al., 2026; Zhang et al., 2026a; Lin et al., 2026a).

A main direction of harness evolution is to identify reusable experience from past interactions and incorporate it into agent harnesses, often in the form of rules, prompts, or skills (Alzubi et al., 2026; Zhang et al., 2026b; Yang et al., 2026; Agrawal et al., 2026). While these approaches yield improvements in many scenarios, they are less effective in long-horizon tasks (Figure 1), where overlong context remains a major bottleneck (Liu et al., 2024; Bai et al., 2025; Min et al., 2026). In this setting, long interactions gradually accumulate redundant information, hindering effective skill invocation and degrading skill quality.

Previous studies have also explored approaches for context compression, organization, and memory management (Hu et al., 2025; Kang et al., 2026; Zhou et al., 2026). Yet context management typically remains fixed or is only slightly adjusted even as the harness evolves, limiting adaptation across task environments (Yao et al., 2026; Li et al., 2026c; Zhang et al., 2026a; Lee et al., 2026). This raises a critical question: **can long-horizon agents evolve their context policies automatically, systematically, and effectively?**

Evolving context policies, however, introduces unique challenges beyond existing skill or rule evolution approaches. Context policies govern how information is presented, organized, and shared throughout task execution, making their effects hard to identify and, consequently, hard to optimize. The first challenge is **failure attribution**: a failed trajectory does not directly reveal whether the problem comes from the model's reasoning, the environment, or the way information is managed, as these factors are often intertwined (Zhang et al., 2026a; Li et al., 2026a). The second challenge is **target identification**: even when context management is identified as a source of failure, it remains unclear which part of the context policy should be evolved (Lee et al., 2026; Ma et al., 2026; Li et al., 2026a).

We introduce ContextEvo, a framework for evolving context policies across the full information lifecycle of long-horizon agent execution. From execution trajectories, ContextEvo reconstructs how information enters, persists, and reaches the model. It then uses cross-case evidence and counterfactual replay to determine whether a failure results from context management. When supported, this attribution is translated into a constrained policy update. By jointly evolving *Input Assembly*, *History Maintenance*, and *Context Orchestration*, ContextEvo extends harness evolution from reusable rules, prompts, and skills to systematic management of information throughout the task.

We evaluate ContextEvo on three long-horizon benchmarks. Starting from Pi-agent, ContextEvo improves full-benchmark scores by 3.1–4.4 percentage points under DeepSeek-V4-Flash, and the evolved policies achieve competitive performance compared to other agent harnesses. Further analysis shows that the policy updates adapt to the information demands of each environment. Compared with existing methods that rely on fixed or locally adapted context-management strategies, ContextEvo achieves stronger performance by evolving the context policy across the full information lifecycle.

Our key contributions are as follows:

- We identify full-lifecycle context management as a critical target for long-horizon harness evolution, where existing evolution methods can struggle as context grows.
- We formulate context management as an evolvable full-lifecycle policy and develop ContextEvo, which uses evidence-based diagnosis and counterfactual validation to localize context failures and perform constrained policy updates.
- Across three long-horizon benchmarks, ContextEvo improves full-benchmark performance and produces policy updates that adapt to environment-specific information demands.

## 2 Related Work

**Agent Performance on Long-Horizon Tasks.** In long-horizon tasks, agents continuously interact with the environment and accumulate information throughout execution (Liu et al., 2024; Bai et al., 2025; Min et al., 2026). Previous work has improved agents through reinforcement learning (Chen

et al., 2025; Xi et al., 2026) and through the design of the agent harness, which organizes execution and controls what information the model receives across steps (Lee et al., 2026; Ma et al., 2026). At the harness level, methods use plans and environmental feedback to guide execution (Erdogan et al., 2025; Zhou et al., 2024a), or manage interaction histories to preserve relevant information (Hu et al., 2025; Wan et al., 2026; Kang et al., 2026). Yet a harness designed in advance may not meet all the needs that arise during a long-horizon task. We therefore study how context management within the harness can evolve from execution trajectories.

**Context Management for Complex Tasks.** In complex real-world tasks, decisions often depend on goals, constraints, and observations encountered many steps earlier, while new interactions continually add to the history. Context management is therefore critical to keeping relevant information available when it is needed. Some methods organize or summarize the history around subgoals and completed steps (Hu et al., 2025; Wu et al., 2025; Ye et al., 2026). Others train agents to maintain compact task states (Zhou et al., 2026; Li et al., 2026b) or to choose memory operations during execution (Zhang et al., 2026e; Yu et al., 2026; Lu et al., 2026). External managers and persistent stores provide another way to retain information and present it when needed (Yi et al., 2026; Li et al., 2026c; Wu et al., 2026; Lin et al., 2026b). ACON (Kang et al., 2026) and TRACE (Min et al., 2026) further use execution feedback to refine context-compression rules. Rather than optimizing a single memory operation or compression policy, CONTEXTEVO uses execution trajectories to reveal deficiencies in the context-management mechanism and revise the harness: what enters the context, how information is retained or transformed, and which models receive it.

**Self-Improving Agents and Harness Evolution.** Self-improving agents use execution experience to update reusable playbooks, skills, memory, and other harness components (Zhang et al., 2026d;c;a; Lin et al., 2026a; Wei et al., 2026; Jiang et al., 2026; Xu et al., 2026; Li et al., 2026a). Whether these updates help on later tasks also depends on how past experience is retrieved and used (Hu et al., 2026; Palmeira Ferraz et al., 2026). In long-horizon tasks, growing histories and irrelevant content can bury important information within the context (Wu et al., 2025; Kang et al., 2026). Even a useful skill may therefore have little effect if the agent cannot use it when needed. Failure-analysis methods examine trajectories to locate critical errors (Barke et al., 2026; Zhu et al., 2025; Qi et al., 2026), while Causal Agent Replay (Shah, 2026) tests the effects of changes to individual steps. Despite these advances, it remains unclear how to systematically evolve context policies from trajectory-level evidence across the full information lifecycle.

## 3 PRELIMINARIES

This section introduces the setting of agent harness evolution and defines the *context policy* studied in this work. We first describe a general harness evolution setting, where the harness evolves while keeping the base model fixed. We then focus on harness context management and define context policy to specify how information is presented, organized, and shared during agent execution.

### 3.1 AGENT HARNESS EVOLUTION

We consider an agent as a system composed of a base model $M$ and a harness $H$. The base model is responsible for decision making, while the harness defines the execution procedure for the model, including interaction with external tools, environments, and other agents. More importantly, the harness determines the context upon which the model's decisions rely.

In this work, we focus on harness-level evolution: the base model $M$ remains fixed, while the harness $H$ evolves to improve agent performance.

### 3.2 CONTEXT POLICY

Since the harness determines the context provided to the model during execution, we study harness evolution from the perspective of context management. Within the harness, we define a ***context policy*** $P$ that governs how information is presented, maintained, and shared through the harness. At time $t$, let $\mathcal{I}_{\leq t}$ denote the information that has entered the system up to that point, including task instructions, observations, tool outputs, and intermediate information. The policy produces the

model-visible context $c_t = P(\mathcal{I}_{\leq t})$, with

$$a_t \sim M(\cdot \mid c_t). \tag{1}$$

A context policy is characterized along three dimensions: *Input Assembly*, *History Maintenance*, and *Context Orchestration*. Table 3.2 introduces these dimensions and common realizations in agent harnesses. The policy space is not limited to these realizations and allows other implementations and combinations across dimensions. Together, these dimensions define an abstraction of the full information lifecycle without depending on particular harness components.

| **Dimension** | **Role** | **Common realizations** |
|---|---|---|
| Input Assembly | Context entry preparation | Tool-result formatting; observation compression |
| History Maintenance | Historical context management | Compaction; summarization; persistent state |
| Context Orchestration | Context access and sharing across agents and subtasks | Instructions and tools for context access; subagent instructions and feedback design |

Table 1: Context-policy dimensions and common realizations in agent harnesses.

Formally, we represent a context policy as

$$P = \left(P^{\text{in}}, P^{\text{hist}}, P^{\text{orch}}\right) \in \mathcal{P}, \tag{2}$$

where $\mathcal{P}$ denotes the design space of harness context policies, and the three components correspond to the three dimensions above. During evolution, $P$ can be updated by modifying, combining, or introducing implementations along these dimensions.

### 3.3 Evolution Objective

The goal of context-policy evolution is to identify a policy that improves agent performance under a fixed base model $M$:

$$P^* = \arg\max_{P \in \mathcal{P}} \mathbb{E}_{\tau \sim (M, H_P)}[R(\tau)], \tag{3}$$

where $R(\tau)$ denotes the task-level reward. The notation $\tau \sim (M, H_P)$ denotes an execution under harness $H_P$ with the context policy $P$.

## 4 Method

Under the evolution objective in Equation 3, ContextEvo uses long-horizon execution trajectories to evolve a context policy while keeping the base model $M$ fixed. It addresses the two challenges identified above: determining whether a failure arises from how the harness manages information exposed to the model, and identifying which policy dimension should be evolved.

During evolution, ContextEvo first reconstructs evidence about how information is handled across observed trajectories. It then uses this evidence across cases to attribute failures to context management and identify the corresponding policy dimensions. Supported attributions are used to update the context policy by modifying existing implementations, combining them, or introducing new ones along the identified dimensions.

We describe evidence reconstruction, failure attribution, and policy update in Sections 4.1–4.3, respectively. Within one evolution run, the acting agent, analysis agent, replay calls, and policy editor use the same fixed base model $M$.

### 4.1 Trajectory Evidence Reconstruction

*Evidence Reconstruction* establishes the evidence for subsequent failure attribution by recovering how information is exposed and carried through each trajectory. It combines a trajectory-wise information-flow scan with further review of the model-visible context.

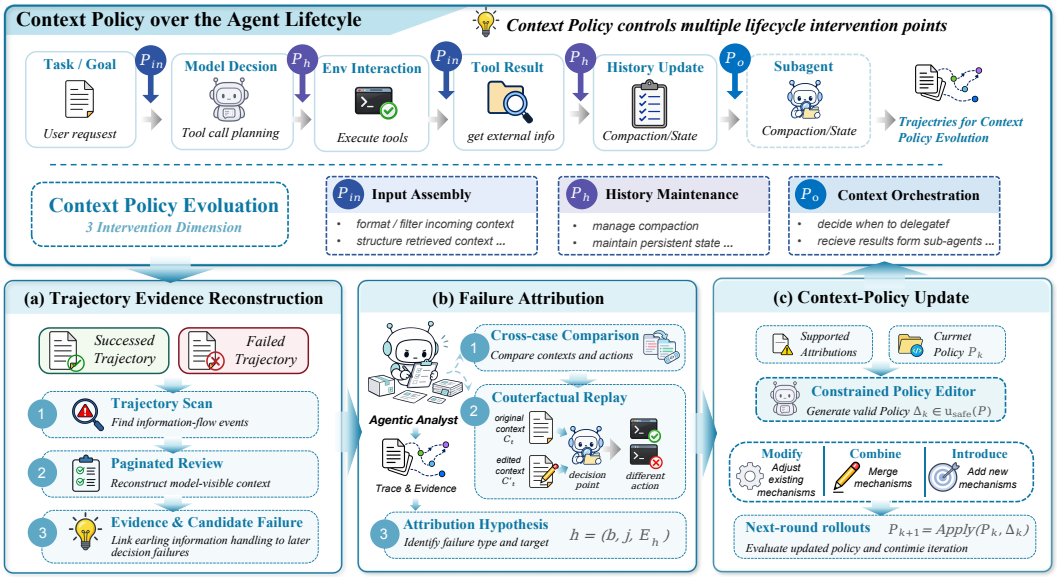



Figure 2: **The overview of CONTEXTEVO.** The context policy governs how information is presented, organized, and shared through the agent harness while the base model remains fixed. CONTEXTEVO reconstructs trajectory evidence, attributes context-management failures to policy dimensions, and applies constrained updates for the next evolution round.

**Trajectory scan.** Information relevant to a later decision may have appeared many steps earlier, and long trajectories may exceed a single analysis context. We therefore scan the full trajectory for a compact record of information-processing events across the execution, including when task-relevant information is presented, organized, or revisited, as well as task progress and repeated reads or executions.

**Paginated review.** The extracted record alone does not determine how information handling contributed to a failure. We therefore review the trajectory in successive segments, which we refer to as *pages*, while reconstructing the model-visible context at relevant decision points. Throughout the review, the reviewer maintains a running *note* that records task progress, unresolved events, and relevant context changes, updating it after each page. The note carries information across pages, allowing the review to relate earlier information-processing events to later decisions. The review identifies candidate context-management failures while distinguishing failures that may instead arise from the model, tools, or environment.

### 4.2 FAILURE ATTRIBUTION AND TARGET IDENTIFICATION

Given the reconstructed trajectory evidence, CONTEXTEVO determines whether a failure can be attributed to context management and, if so, which policy dimension should be adjusted. An analysis agent then examines candidate decision points, compares evidence across successful and failed trajectories, and uses counterfactual replay when the observed evidence does not resolve the attribution.

**Counterfactual replay.** Rerunning the full task is both costly and difficult to interpret, since later execution may diverge for reasons unrelated to the context change being tested. We instead hold the execution prefix $\tau_{<t}$ fixed and modify only the model-visible context at decision $t$,

$$\widetilde{c}_t = \mathrm{Edit}(c_t).$$

We then compare samples from the original and modified contexts:

$$a_t^{(r)} \sim M(\cdot \mid c_t), \qquad \widetilde{a}_t^{(r)} \sim M(\cdot \mid \widetilde{c}_t), \qquad r = 1, \ldots, N. \tag{4}$$

This local comparison tests whether the proposed context-policy update improves the model's decision at the same execution point without side effects for subsequent actions.

| Harness / policy | Long-Horizon TB | | | DeepSWE | | | BrowseComp-Plus | | |
|---|---|---|---|---|---|---|---|---|---|
| | Held-in (%) | Held-out (%) | All (%) | Held-in (%) | Held-out (%) | All (%) | Held-in (%) | Held-out (%) | All (%) |
| *Harness (DeepSeek-V4-Flash)* | | | | | | | | | |
| OpenCode | 36.1 | 40.3 | 38.5 | 62.5 | 66.2 | 64.6 | **92.1** | 77.6 | 83.9 |
| Codex | 41.8 | 40.1 | 40.8 | 45.8 | 63.1 | 55.8 | 84.2 | 84.5 | 84.4 |
| OpenClaw | 36.9 | **45.9** | 42.0 | 45.8 | 60.0 | 54.0 | 89.5 | **88.8** | **89.1** |
| **Pi-agent (base)** | 43.3 | 40.4 | 41.7 | 54.2 | 72.3 | 64.6 | 81.6 | 82.7 | 82.2 |
| *ContextEvo (DeepSeek-V4-Flash)* | | | | | | | | | |
| **iter-1** | **47.0** (+3.7) | 43.1 (+2.7) | 44.8 (+3.1) | 60.4 (+6.3) | 75.4 (+3.1) | 69.0 (+4.4) | 90.8 (+9.2) | 81.6 (-1.0) | 85.6 (+3.4) |
| **iter-2** | 45.7 (-1.3) | 45.0 (+1.9) | **45.3** (+0.6) | **66.7** (+6.3) | **75.4** (+0.0) | **71.7** (+2.7) | **92.1** (+1.3) | 85.7 (+4.1) | 88.5 (+2.9) |
| *Harness (GPT-5.6-Luna)* | | | | | | | | | |
| OpenCode | 32.0 | 32.3 | 32.1 | 22.9 | 21.5 | 22.1 | 35.5 | 34.7 | 35.1 |
| Codex | **36.3** | **38.6** | **37.6** | **43.8** | **44.6** | **44.3** | **64.0** | **57.1** | **60.1** |
| OpenClaw | 20.5 | 17.4 | 18.8 | 10.4 | 10.8 | 10.6 | 22.4 | 16.3 | 19.0 |
| **Pi-agent (base)** | 31.8 | 29.2 | 30.3 | 6.3 | 18.5 | 13.3 | 35.5 | 39.8 | 37.9 |
| *ContextEvo (GPT-5.6-Luna)* | | | | | | | | | |
| **iter-1** | 27.6 (-4.2) | 33.7 (+4.5) | 31.0 (+0.7) | 6.3 (+0.0) | 15.4 (-3.1) | 11.5 (-1.8) | 46.1 (+10.5) | 33.7 (-6.1) | 39.1 (+1.1) |
| **iter-2** | 33.7 (+6.1) | 34.8 (+1.0) | 34.3 (+3.3) | 14.6 (+8.3) | 13.8 (-1.5) | 14.2 (+2.7) | 43.4 (-2.6) | 40.8 (+7.1) | 42.0 (+2.9) |

Table 2: **Performance of baseline harnesses and CONTEXTEVO over two evolution iterations.** Scores are reported on the evolution (Held-in), evaluation (Held-out), and full task sets. Values in parentheses indicate percentage-point changes from the preceding baseline (Pi-agent for iter-1 and iter-1 for iter-2). Bold and underline indicate the best and second-best scores, respectively.

**Attribution hypothesis.** The analysis produces a candidate attribution hypothesis together with the evidence supporting it. We represent an attribution hypothesis as

$$h = (b, j, \mathcal{E}_h), \tag{5}$$

where $b \in \{\text{missing}, \text{degraded}, \text{buried}, \text{polluted}\}$ characterizes the information failure, $j \in \{\text{in}, \text{hist}, \text{orch}\}$ identifies the policy dimension to which the failure is attributed, and $\mathcal{E}_h$ collects the supporting evidence. This evidence may include decision-point observations, cross-case comparisons, context-growth comparisons, replay results, and relevant counterexamples or alternative explanations.

We retain an attribution only when the evidence distinguishes failed from successful executions and supports the proposed failure type and policy target. Supported attributions are then passed to the policy update stage.

### 4.3 CONTEXT-POLICY UPDATE

Given the supported attributions and the current context policy $P_k$, the editor constructs a constrained update $\Delta_k$. Consistent with the policy space defined in Section 3, the update may modify, combine, or introduce implementations along the dimensions, subject to fixed safety constraints. The context policy is updated as

$$P_{k+1} = \text{Apply}(P_k, \Delta_k), \qquad \Delta_k \in \mathcal{U}_{\text{safe}}(P_k). \tag{6}$$

Here $\mathcal{U}_{\text{safe}}(P_k)$ denotes the updates allowed by these constraints. If the available evidence does not support an update, $\Delta_k = \emptyset$ and $P_{k+1} = P_k$. The updated policy is then used to generate trajectories for the next evolution round, while held-out tasks are reserved for final evaluation of the selected policy $P^*$.

## 5 EXPERIMENTS AND RESULTS

### 5.1 EXPERIMENTAL SETUP

**Tasks and evaluation.** We evaluate CONTEXTEVO in three long-horizon benchmark settings. Long-Horizon-Terminal-Bench (LHTB) (Li et al., 2026d) contains 46 terminal workflows,

DeepSWE (Huang et al., 2026) contains 113 repository-level software-engineering tasks, and BrowseComp-Plus (Chen et al., 2026) uses a hard, search-intensive subset of 174 questions. LHTB uses a dense task score, while DeepSWE and BrowseComp-Plus use binary task scores from verifiers. For each benchmark, we report results on the full task set and separately on the evolution set (Held-in) and reserved evaluation set (Held-out). Benchmark distributions and details, along with the construction and motivation of the BrowseComp-Plus subset are detailed in Appendix A.

**Baselines and evolution settings.** We evaluate four agent harness baselines, i.e., OpenCode, Pi-agent, Codex, and OpenClaw, with two different models: DeepSeek-V4-Flash-0731 and GPT-5.6-Luna. We implement CONTEXTEVO in the open-source Pi-agent harness and run two successive evolution iterations to assess whether performance continues to improve beyond the first iteration. Within each benchmark-model setting, we keep the base model used for failure attribution and target identification fixed across evolution iterations. The evolution pipeline uses only held-in trajectories, and each evolved policy and its corresponding baseline are evaluated on the same task split under identical execution settings. Model versions, inference parameters, and harness configurations are provided in Appendix A.2.

### 5.2 MAIN RESULTS

Table 5.1 compares the initial baseline and the CONTEXTEVO policies with established harnesses.

**CONTEXTEVO is effective across different base models and task settings, while remaining highly competitive with prominent harnesses.** After evolution, ContextEvo achieves the best full-benchmark scores on LHTB and DeepSWE under DeepSeek-V4-Flash, reaching 45.3% and 71.7%, respectively, and scores 88.5% on BrowseComp-Plus, only 0.6% below OpenClaw. Under GPT-5.6-Luna, the evolved policy also improves over the initial baseline on all three benchmarks and ranks second only to the Luna-native Codex harness on LHTB and BrowseComp-Plus, with scores of 34.3% and 42.0%, respectively. These results show that CONTEXTEVO is broadly effective across models and task settings while achieving performance comparable to established and strong harnesses.

**CONTEXTEVO enables sustained gains across successive evolution iterations.** On the full task sets, iter-2 improves over iter-1 in all six benchmark–model settings. Under DeepSeek-V4-Flash, the scores increase from 44.8% to 45.3% on LHTB, from 69.0% to 71.7% on DeepSWE, and from 85.6% to 88.5% on BrowseComp-Plus. The same trend holds under GPT-5.6-Luna, where the scores increase from 31.0% to 34.3%, from 11.5% to 14.2%, and from 39.1% to 42.0%, respectively. These consistent second-round gains indicate that CONTEXTEVO can continue refining the context policy beyond the initial update rather than producing only a one-time improvement.

| Agent | Reward ↑ | Tokens (M) ↓ |
|---|---|---|
| Baseline (Pi) | 0.417 | 8.57 |
| Skill-only | 0.370 | 9.61 |
| CONTEXTEVO | **0.448** | **7.52** |
| CONTEXTEVO + skill | 0.419 | 9.69 |

Table 3: **Context-policy vs. skill evolution on LHTB-46.** Reward and mean token usage per run are reported for different settings.

## 6 ANALYSIS

### 6.1 BEYOND SKILL EVOLUTION IN LONG-HORIZON TASKS

**Context-policy evolution outperforms skill-only evolution in the long-horizon setting.** Following recent trajectory-driven harness evolution (Zhang et al., 2026a), we derive procedural skills from the same held-in LHTB tasks used by CONTEXTEVO. Across 46 tasks, skill-only reward falls from 0.417 to 0.370, while mean tokens rise from 8.57M to 9.61M (Table 4). CON-

| Agent | Reward ↑ | Tokens (M) ↓ |
|---|---|---|
| Baseline (Pi) | 0.417 | 8.57 |
| Skill-only | 0.370 | 9.61 |
| CONTEXTEVO | **0.448** | **7.52** |
| CONTEXTEVO + skill | 0.419 | 9.69 |

Table 4: LHTB task reward and mean tokens per run.

TEXTEVO reaches 0.448 with 7.52M tokens;
adding skills to it yields 0.419 with 9.69M.

The trajectories reveal three limits of skill evolution. **(1) Invocation gaps.** In long-horizon tasks, complex contexts make skills harder to use at the right decision points: only 20 of 44 inspectable skill-only trajectories read a skill body. **(2) Local goal drift.** Even when used, skill guidance can divert long-horizon execution toward a local objective. In the DuckDB case, the agent uses the skill to pass local checks, but its task objective shifts and it does not continue refining the final patch. **(3) Partial workflow coverage.** A skill evolved from long-horizon trajectories may cover only part of a larger workflow. In the MODFLOW audit, the agent follows a generic benchmark-loop skill and completes the artifacts requested by the skill, but the final submission remains incomplete relative to the requirements of the full workflow. Together, these patterns motivate a full-lifecycle context policy that keeps task evidence available from initial observations through final delivery. Additional skill-only cases and analyses are provided in Appendix C.

### 6.2 CONTEXT MANAGEMENT BASELINES

| Method | DeepSWE | LHTB | BrowseComp-Plus |
|---|---|---|---|
| *Base agent* | | | |
| Pi-agent base | 0.133 | 0.303 | 0.379 |
| *Static context management* | | | |
| ReSum | 0.062 | 0.266 | 0.385 |
| ACON | 0.124 | 0.262 | 0.374 |
| *Dynamic context management* | | | |
| TACO | 0.115 | 0.281 | 0.374 |
| *Full context-policy evolution* | | | |
| **CONTEXTEVO** | **0.142** | **0.343** | **0.420** |

Table 5: Context-management baselines

Following the skill-evolution comparison, we compare static compression (ReSum and ACON) (Wu et al., 2025; Kang et al., 2026), local compression evolution (TACO) (Ren et al., 2026), and full context-policy evolution (CONTEXTEVO). All conditions use GPT-5.6-Luna under matched Pi-agent protocols.

CONTEXTEVO achieves the highest score on all three benchmarks: 0.142 on DeepSWE, 0.343 on LHTB, and 0.420 on BrowseComp-Plus. This indicates that full-policy evolution better addresses long-horizon context failures than static compression or local adjustment alone. Additional cases appear in Appendix C.

### 6.3 LONG-HORIZON ENVIRONMENT-SPECIFIC POLICY EVOLUTION

Across the three benchmarks, task environments impose different context pressures. Figure 3 shows that CONTEXTEVO learns different context-policy realizations from these pressures: selective observation handling and persistent execution state for LHTB, specification and verification continuity for DeepSWE, and recoverable evidence organization for BrowseComp-Plus. Thus, CONTEXTEVO adapts the context policy to each task environment rather than applying one fixed strategy across tasks. The appendix provides additional case details and analysis, including examples where policy evolution introduces context-management mechanisms absent from the initial harness.

These learned policies transfer across models within the same environment, but not reliably across environments. The LHTB policy improves both DeepSeek and Luna, whereas applying the frozen LHTB policy to DeepSWE decreases performance for both models and produces mixed effects on BrowseComp-Plus (Figure 3). This shows that policies can be reused across models when the context pressure is shared, while environments with substantially different pressures require renewed policy evolution.

### 6.4 COMPONENT ABLATION

We next evaluate the three-stage evolution pipeline on LHTB by comparing the native Pi-agent pipeline, the complete ContextEvo pipeline, and leave-one-operator-out variants under the same DeepSeek-V4-Flash-0731 protocol. Table 6 reports results on the held-in and held-out splits.

**The complete evolution pipeline achieves the strongest overall performance.** Full reaches 44.78%, improving over Native at 41.66% by 3.12 percentage points. **Evidence Reconstruction is the most consequential stage.** Removing Paginated Trajectory Review lowers the overall score to 32.64% ($-12.15$ pp) and the held-out score to 26.19% ($-16.93$ pp). **Failure Attribution**

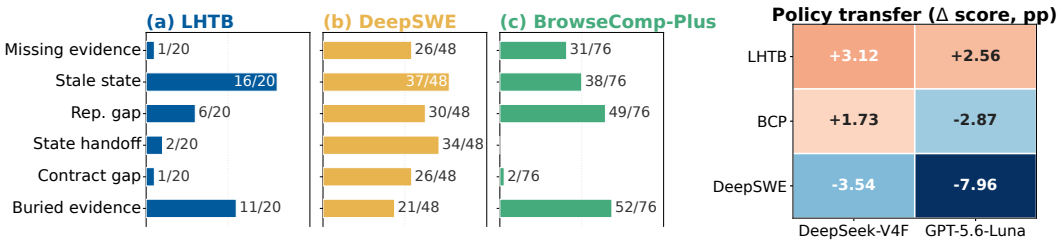



Figure 3: **Environment-specific context problems and policy transfer.** Bars show the share of audited cases with each multi-select context problem; labels report counts. The heatmap shows signed score changes (percentage points) from applying the LHTB-evolved policy under DeepSeek and Luna.

| Variant | Held-in | Held-out | All | Δ vs. Full |
|---|---|---|---|---|
| Native Pi-agent | 43.30% | 40.39% | 41.66% | −3.13 pp |
| Full ContextEvo | 46.95% | 43.12% | **44.78%** | – |
| w/o Paginated Trajectory Review | 41.01% | 26.19% | 32.64% | −12.15 pp |
| w/o Cross-Case Aggregation | 40.46% | 44.05% | 42.49% | −2.29 pp |
| w/o Counterfactual Replay | 48.57% | 35.29% | 41.06% | −3.72 pp |
| w/o Context-Policy Update | 32.57% | 42.78% | 38.34% | −6.44 pp |

Table 6: LHTB-46 component ablations with DeepSeek-V4-Flash-0731 under the matched Pi-agent protocol; Δ denotes the all-task difference from Full.

**and Context-Policy Update provide complementary gains.** Removing Cross-Case Aggregation, Counterfactual Replay, or Context-Policy Update lowers the overall score by 2.29, 3.72, and 6.44 percentage points, respectively; the replay ablation is especially damaging on held-out tasks, reducing the score from 43.12% to 35.29%. Together, these results show that the strongest performance comes from the complete evolution loop.

**Cross-harness validation.** To check that the gains are not confined to Pi-agent, we also evaluate ContextEvo on OpenCode and OpenClaw with GPT-5.6-Luna on BrowseComp-Plus hard174. ContextEvo improves OpenCode from 35.06% (61/174) to 40.80% (71/174), a gain of 10 tasks (+5.74 pp), and OpenClaw from 18.97% (33/174) to 45.40% (79/174), a gain of 46 tasks (+26.44 pp). The matched protocol and full comparison are reported in Appendix C.4.

## 7 Conclusion

We presented ContextEvo, a framework that evolves a context policy for long-horizon agent execution while keeping the base model fixed within each benchmark–model run. The policy spans three dimensions—Input Assembly, History Maintenance, and Context Orchestration—and is updated through evidence reconstruction, failure attribution and target identification, and constrained policy update.

Across the reported benchmark and model settings, the selected policies improve several full-benchmark point estimates over the Pi-agent baseline, while held-out changes vary by setting. Case-level analysis indicates that the reviewed updates emphasize different information demands in terminal workflows, repository engineering, and multi-hop search.

The evidence also defines the limits of the method's claims. Cross-case evidence supports an attribution hypothesis, bootstrap evidence can justify a bounded update, and decision-point replay measures only local next-action sensitivity. These forms of evidence do not by themselves establish a task-level causal reward contribution for an individual policy mechanism.

## AI USE STATEMENT

In this work, we used generative AI tools to assist with result interpretation, literature organization, and the drafting, editing, and formatting of text, tables, and figures. The authors reviewed and verified all AI-assisted outputs, including claims, citations, analyses, and presentation, and take responsibility for the final content of this work.

## REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026. URL https://arxiv.org/abs/2603.02766.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026. URL https://arxiv.org/abs/2602.02475. Accepted to EMNLP Findings 2026.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025. URL https://arxiv.org/abs/2502.01600.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 15419–15462. PMLR, 2025. URL https://proceedings.mlr.press/v267/erdogan25a.html.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1575. URL https://aclanthology.org/2025.acl-long.1575/.

Qisheng Hu, Quanyu Long, and Wenya Wang. When continual learning moves to memory: A study of experience reuse in llm agents. *arXiv preprint arXiv:2604.27003*, 2026. URL https://arxiv.org/abs/2604.27003.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026. URL https://arxiv.org/abs/2607.07946.

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. Harnessevolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 2026. URL https://arxiv.org/abs/2609.00829.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon llm agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.00615. Accepted at ICML 2026.

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

Laizhen Li, Jiarui Li, Juanjuan Zhao, Kejiang Ye, Ye Li, Cheng-Zhong Xu, and Xitong Gao. Grow the harness, not the context: From strategy-free scaffolds to reusable specialist agents. *arXiv preprint arXiv:2609.26760*, 2026a. URL https://arxiv.org/abs/2609.26760.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. MemPO: Self-memory policy optimization for long-horizon agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23286–23301, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1166. URL https://aclanthology.org/2026.findings-acl.1166/.

Xiaochuan Li, Ryan Ming, Meng Chu, Shuai Shao, Rong Jin, and Chenyan Xiong. ACM: Agentic context management for long horizon tasks. *arXiv preprint arXiv:2607.23809*, 2026c. URL https://arxiv.org/abs/2607.23809.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*, 2026d. URL https://arxiv.org/abs/2607.08964.

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

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic rl via end-to-end optimized context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21074–21125, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.966. URL https://aclanthology.org/2026.acl-long.966/.

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

Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability. *arXiv preprint arXiv:2608.06503*, 2026. URL https://arxiv.org/abs/2608.06503.

Thomas Palmeira Ferraz, Romain Deffayet, Vassilina Nikoulina, Hervé Déjean, and Stéphane Clinchant. Retrieval-augmented llm agents: Learning to learn from experience. *arXiv preprint arXiv:2603.18272*, 2026. URL https://arxiv.org/abs/2603.18272. Accepted at EMNLP 2026 (Main Conference).

Yunjia Qi, Zehua Yin, Xintong Shi, Hao Peng, Songyuanyi Lu, Yixian Liu, Richeng Xuan, Yuhong Liu, Zhichao Hu, Xiaozhi Wang, Lei Hou, Bin Xu, and Juanzi Li. TRAJDEBUG: Tracing error lifecycle to identify critical failures in long-horizon agent trajectories. *arXiv preprint arXiv:2608.06346*, 2026. URL https://arxiv.org/abs/2608.06346.

Jincheng Ren, Siwei Wu, Yizhi Li, Kang Zhu, Shu Xu, Boyu Feng, Ruibin Yuan, Wei Zhang, Riza Batista-Navarro, Jian Yang, and Chenghua Lin. A self-evolving framework for efficient terminal agents via observational context compression. *arXiv preprint arXiv:2604.19572*, 2026. URL https://arxiv.org/abs/2604.19572.

Jaineet Shah. Causal agent replay: Counterfactual attribution for llm-agent failures. *arXiv preprint arXiv:2606.08275*, 2026. URL https://arxiv.org/abs/2606.08275.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. COMPASS: Enhancing agent long-horizon reasoning with evolving context. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3360–3380, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.152. URL https://aclanthology.org/2026.acl-long.152/.

Tianxin Wei, Zhan Shi, Minhua Lin, Bing He, Zewen Liu, Yisi Sang, Yuanchen Bei, Xuying Ning, Jiaru Zou, Ting-Wei Li, Xiao Lin, Yanjun Zhao, Chi Wang, Benoit Dumoulin, Dakuo Wang, Jingrui He, and Hanqing Lu. Evo-harness: Context-to-harness skill compilation for self-evolving agents. *arXiv preprint arXiv:2608.15071*, 2026. URL https://arxiv.org/abs/2608.15071. Accepted at EMNLP 2026 (Main Conference).

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

Yifan Wu, Lizhu Zhang, Yuhang Zhou, Mingyi Wang, Bo Peng, Serena Li, Xiangjun Fan, and Zhuokai Zhao. Remember when it matters: Proactive memory agent for long-horizon agents. *arXiv preprint arXiv:2607.08716*, 2026. URL https://arxiv.org/abs/2607.08716.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: An open-source framework to train llm agents for long-horizon decision making via multi-turn rl. In *International Conference on Learning Representations*, volume 2026, pp. 12685–12727,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Jinghan Xu, Yikai Zhang, Aili Chen, Weiyuan Li, Jiaqing Liang, and Deqing Yang. Verify smarter, evolve further: Efficient harness evolution through behavior-aware verification. *arXiv preprint arXiv:2608.27311*, 2026. URL https://arxiv.org/abs/2608.27311.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026. URL https://arxiv.org/abs/2605.23904.

Yilun Yao, Shan Huang, Elsie Dai, Zhewen Tan, Zhenyu Duan, Shousheng Jia, Yanbing Jiang, and Tong Yang. ARC: Active and reflection-driven context management for long-horizon information seeking agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18644–18659, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.930. URL https://aclanthology.org/2026.findings-acl.930/.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Siheng Chen, and Yong Jiang. AgentFold: Long-horizon web agents with proactive context folding. In *International Conference on Learning Representations*, volume 2026, pp. 154754–154776, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/fa948624dfde013671e72c1a7ca4aebc-Paper-Conference.pdf.

Lu Yi, Runlin Lei, Liuyi Yao, Yuexiang Xie, Yuyang Li, Wenhao Zhang, Zhewei Wei, Yaliang Li, and Jian-Yun Nie. Learning agent-compatible context management for long-horizon tasks. *arXiv preprint arXiv:2605.30785*, 2026. URL https://arxiv.org/abs/2605.30785.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

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

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a. URL https://arxiv.org/abs/2606.09498.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026b. URL https://arxiv.org/abs/2604.01687. Accepted at the Conference on Language Modeling (COLM 2026).

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026c. URL https://arxiv.org/abs/2602.02474.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y. Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language

models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026d. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8a94ff6f922d995d7d3f4ebf4143e442-Paper-Conference.pdf.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19149–19164, San Diego, California, United States, 2026e. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.956. URL https://aclanthology.org/2026.findings-acl.956/.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62138–62160. PMLR, 2024a. URL https://proceedings.mlr.press/v235/zhou24r.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, volume 2026, pp. 58413–58438, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5fc8b3bdfbb9167b5144df5d3fae4616-Paper-Conference.pdf.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025. URL https://arxiv.org/abs/2509.25370.

Table 7: Benchmark inventories and fixed evolution splits. Held-in tasks provide trajectories for evidence reconstruction, attribution, replay, and policy updating; held-out tasks are reserved for evaluation.

| Benchmark | Scenario | All | Held-in | Held-out |
|---|---|---|---|---|
| LHTB | terminal workflows | 46 | 20 | 26 |
| DeepSWE | repository software engineering | 113 | 48 | 65 |
| BrowseComp-Plus | multi-hop deep research | 174 | 76 | 98 |

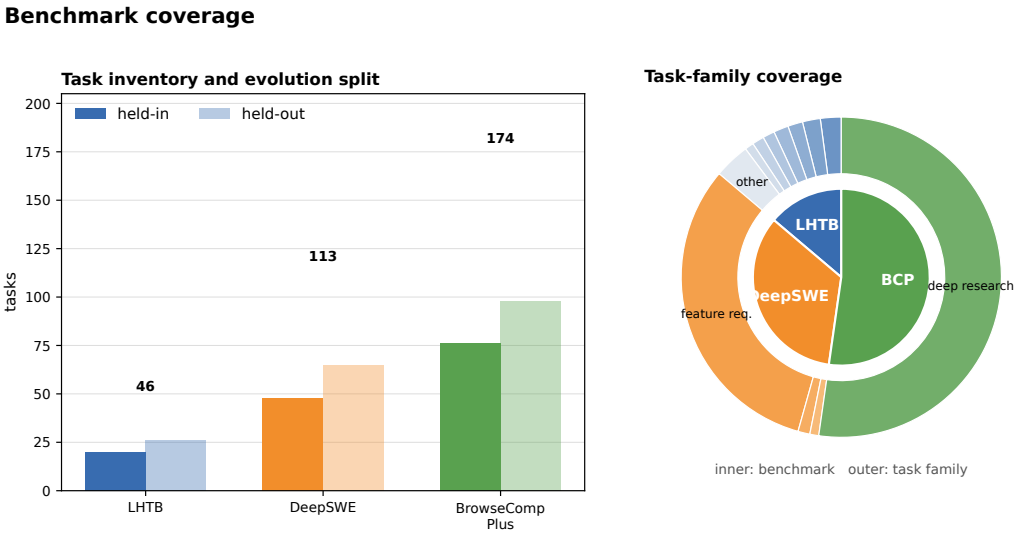



Figure 4: **Benchmark coverage.** The left panel reports the full inventory and the held-in/held-out split. The sunburst shows the benchmark inventory in the inner ring and task-family composition in the outer ring. LHTB shows its seven largest categories plus an "other" bucket, DeepSWE shows request types, and BrowseComp-Plus is a uniform multi-hop deep-research setting.

# A Benchmark and Experimental Protocol

## A.1 Benchmark settings

**Benchmark profiles.** We evaluate three task settings with different sources of long-horizon context. Long-Horizon-Terminal-Bench (LHTB) contains terminal workflows that span software engineering, games, tool-use, multimodal work, and scientific or systems tasks. DeepSWE contains repository-level software-engineering requests, dominated by feature requests with a smaller set of bug fixes and enhancements. BrowseComp-Plus contains multi-hop deep-research questions that require repeated retrieval and evidence synthesis. Table 7 gives the task counts and fixed evolution split; Figure 4 shows the task-family coverage, while Figure 5 and Table 8 report workload statistics measured from the baseline trajectories.

**BrowseComp-Plus hard subset.** The official BrowseComp-Plus release has 830 questions. We use a 174-question hard subset because the full benchmark produces a ceiling effect for the current frontier model used in our main comparison. We first grade the publicly released o3 and GPT-5 reference answers with the benchmark judge, retain questions that both models miss, and then require at least 30 o3 searches. This yields questions that are both difficult for frontier references and search-intensive. We apply `random.Random(15)` to the sorted task names and assign the first 76 questions to held-in and the remaining 98 to held-out. The held-out questions are never used for trajectory scan, paginated trajectory review, failure attribution, replay, or policy updating. The complete construction and offline scoring procedure, including answer extraction and infrastructure-error handling, are documented here so that the main text need only state the resulting subset and split.

Table 8: Workload statistics from frozen Pi/DeepSeek-V4-Flash baseline trajectories. Entries are median [p25, p75]; trace tokens estimate analyst-visible text and are not billing tokens.

| Benchmark | Trajectories | Tool calls | Agent turns | Estimated trace tokens |
|---|---|---|---|---|
| LHTB | 44 | 75.0 [52.5, 96.5] | 69.0 [45.0, 87.3] | 80,712 [58,862, 111,707] |
| DeepSWE | 112 | 120.5 [85.8, 158.0] | 103.5 [65.8, 138.5] | 124,205 [92,726, 157,711] |
| BrowseComp-Plus | 174 | 32.5 [17.3, 52.8] | 27.5 [14.0, 45.8] | 70,866 [38,626, 128,303] |

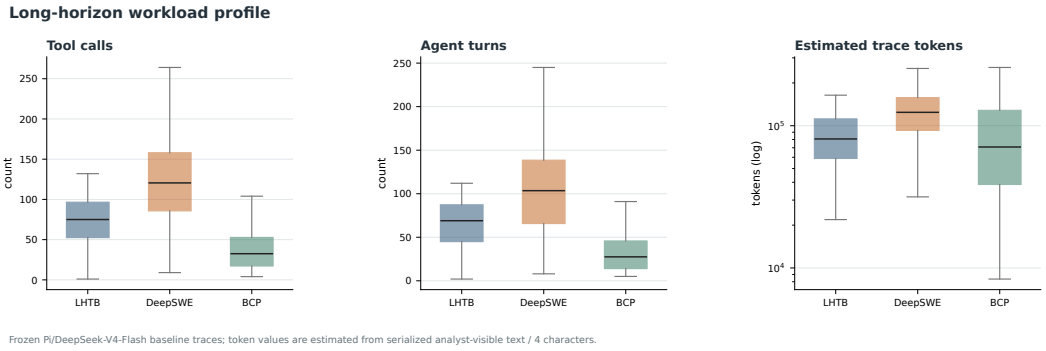



Figure 5: **Long-horizon workload profile.** The three panels show the distributions of tool calls, agent turns, and estimated trajectory text volume in frozen Pi/DeepSeek-V4-Flash baseline traces. The token measure is an audit estimate from serialized analyst-visible text rather than provider billing tokens; the corresponding median and interquartile values are listed in Table 8.

### A.2 Evaluation configuration

**Model configurations.** We use the same model configurations across the three benchmarks. Table 10 records the model release, provider identifier, reasoning setting, context limit, output limit, and generation parameters.

**Harness configurations.** Table 11 records the execution settings for the four harnesses. The task timeout is 5,400 seconds for the reported runs; setup limits and trial parallelism follow the benchmark configurations. Each task is attempted once, and the baseline and self-evolved policy use the same harness settings within a comparison.

**Scoring and result handling.** Let $U$ denote one of the fixed held-in, held-out, or full task subsets. For LHTB, each task receives a partial reward $r_t^{\mathrm{LHTB}} \in [0, 1]$, and the reported score is

$$S_{\mathrm{LHTB}}(U) = \frac{1}{|U|} \sum_{t \in U} r_t^{\mathrm{LHTB}}. \tag{7}$$

For DeepSWE, let $y_t^{\mathrm{DeepSWE}} \in \{0, 1\}$ indicate whether the repository-level task is solved according to the benchmark evaluator. The score is

$$S_{\mathrm{DeepSWE}}(U) = \frac{1}{|U|} \sum_{t \in U} y_t^{\mathrm{DeepSWE}}. \tag{8}$$

For BrowseComp-Plus, let $y_t^{\mathrm{BCP}} \in \{0, 1\}$ indicate whether the submitted answer is judged correct. The score is

$$S_{\mathrm{BCP}}(U) = \frac{1}{|U|} \sum_{t \in U} y_t^{\mathrm{BCP}}. \tag{9}$$

We report these means as percentages. An agent timeout or an execution failure attributable to the agent receives a score of zero, with the timeout limit determined by the corresponding benchmark

Table 9: BrowseComp-Plus hard-subset construction.

| Selection stage | Criterion |
|---|---|
| Reference difficulty | o3 and GPT-5 reference answers are both judged incorrect |
| Long-horizon filter | o3 issues at least 30 searches |
| Resulting subset | 174 questions from the 830-question release |
| Evolution split | seed 15; 76 held-in / 98 held-out |

Table 10: Model configurations used in the experiments.

| Setting | Configuration |
|---|---|
| DeepSeek-V4-Flash-0731 | `openai/deepseek-v4-flash`; reasoning enabled; context 262,144; max output 32,768 |
| GPT-5.6-Luna | `openai/gpt-5.6-luna`; reasoning enabled; context 262,144; max output 32,768 |
| Generation parameters | Temperature and top-$p$ use the provider defaults; no additional sampling override |

configuration. A trial is rerun only when the failure is confirmed to be caused by the evaluation infrastructure; the confirmed rerun result replaces the failed run before the benchmark score is computed.

# B Method Details

## B.1 Context-policy realization

The editor operates over eight context intervention points exposed at the harness–policy boundary. At runtime, the fixed harness adapter maps native execution events to the corresponding lifecycle points and invokes the ContextProgram where applicable. Decisions from these points jointly determine the context exposed to the agent. The table lists the current realization and illustrative policy modifications at each point; an evolution step may leave a point unchanged, span multiple points, or introduce a mechanism that is not present in the current realization. The ContextProgram is the executable realization of the abstract context policy $P$ inside the fixed harness $H$. The eight intervention points are concrete implementation surfaces grouped under the three policy dimensions; an attribution target $j$ identifies a dimension and may therefore map to one or more intervention points.

**Lifecycle interface implementation.** In our implementation, the eight intervention points are exposed through a versioned ContextProgram connected to the agent by a fixed harness adapter. The adapter maps native execution events to the applicable lifecycle points, provides the state available at each point, and applies the resulting context decisions before subsequent execution. Observation handling may preserve an original result or produce a source-linked representation; history handling may transform, summarize, update, or recover earlier evidence; request assembly determines the next model-visible view; and context orchestration controls which evidence crosses execution scopes and how returned results are represented. A point may remain unchanged for a given update, and a policy change may combine behavior across multiple points or introduce a new mechanism. Thus, the intervention-point table instantiates the three dimensions rather than replacing them. $P^{\mathrm{in}}$ governs entry and request assembly, $P^{\mathrm{hist}}$ governs retention and recovery, and $P^{\mathrm{orch}}$ governs cross-scope access and return.

At runtime, the adapter does not execute the eight intervention points as a fixed sequence on every turn. It invokes the points whose lifecycle events are present. A newly produced observation first reaches the observation-entry point; when the accumulated history requires maintenance, the applicable planning, representation-update, memory-update, or recovery operations are invoked before the next model-visible request is assembled. Request assembly then combines the active history with any transformed or recovered evidence. The parent-to-child and child-to-parent points are invoked at their corresponding scope boundaries, so their decisions govern evidence entering a child and evidence returned to its parent. Each component receives the state exposed at its invocation point

Table 11: Harness execution settings. Concurrent trial counts are LHTB/DeepSWE/BCP.

| Harness | Execution configuration |
|---|---|
| OpenCode | v1.15.13-20260715; OpenAI-compatible protocol; context/output 262,144/32,768; task timeout 5,400 s; 46/32/64 concurrent trials |
| Pi-agent | v0.80.6; `openai-responses`; context/output 262,144/32,768; task timeout 5,400 s; 24/32/64 concurrent trials |
| Codex | Frozen comparison release; `openai-chat`; task timeout 5,400 s; 46/32/64 concurrent trials |
| OpenClaw | v2026.6.1; `openai-completions`; context/output 262,144/32,768; task timeout 5,400 s; 46/32/64 concurrent trials |

Table 12: **Context-policy intervention points.** Current realizations and representative policy modifications.

| | Policy interface | Lifecycle intervention point | Current realization | Examples of possible policy modifications |
|---|---|---|---|---|
| 1 | Input Assembly | Observation entry | `observation_projector` | Change whether a newly received observation is retained in its original form or represented by a shorter, source-linked form. |
| 2 | History Maintenance | Compaction planning | `compaction_planner` | Change when historical material becomes eligible for transformation and which complete dependency groups may be selected. |
| 3 | History Maintenance | Historical representation update | `summary_writer` | Change which facts, constraints, validation results, failed approaches, and source references are preserved in a compact representation. |
| 4 | History Maintenance | Evidence memory update | `memory_writer` | Change how durable evidence records are added, replaced, or removed and how their provenance is maintained. |
| 5 | History Maintenance | Evidence recovery | `memory_retriever` | Change when prior evidence is requested and which narrowly relevant records are recovered. |
| 6 | Input Assembly | Request assembly | `context_assembler` | Change how active history, summaries, recovered evidence, and inherited context are selected and ordered. |
| 7 | Context Orchestration | Parent-to-child context dispatch | `child_context_builder` | Change which parent evidence is authorized for an already specified child task. |
| 8 | Context Orchestration | Child-to-parent result return | `child_result_reducer` | Change how child findings, evidence references, uncertainty, and unresolved issues are represented to the parent. |

and returns a context decision that the adapter applies to subsequent execution. Consequently, a policy update may affect one event, several events, or introduce a new intervention without changing the harness's overall execution protocol.

### B.2 CONTEXT-POLICY EVOLUTION

**Evolution data boundary.** For each benchmark and base-model block, an evolution run reads all trajectories in the fixed held-in split for that setting. Held-out tasks are excluded from trajectory scan, paginated trajectory review, failure attribution, replay, policy-update inputs, and candidate selection until the policy is frozen for evaluation. The baseline and the selected policy then use the same task split, model, harness, and runtime protocol. The reported v1 policy starts from the Pi-agent baseline; where v2 is reported, it denotes the next selected policy in the corresponding lineage in Table 5.1. An evolution round may return an empty changeset when the evidence does not support an update, and only the resulting frozen policy $P^*$ is evaluated on held-out tasks. Trajectories from another benchmark or base-model block are not pooled into the same evolution run.

**Deterministic case-card extraction.** Stage 1a runs `tools/case_cards.py` once per trial and reads only ledger requests that carry tool schemas; out-of-band generation calls are not counted as agent decisions. The extractor skips benchmark bookkeeping directories, optionally restricts trials with the fixed split file, and can replace rate-limited BrowseComp-Plus inline rewards with the offline `final_scores.json` values. It forms event-aligned chunks at compaction/context-shrink points and the first edit or write, with a 25-call fallback boundary. Each card records entry statistics (tool-output counts, characters, and observations at least 8,192 characters), retention statistics (context composition, duplicate share, and compactions), distribution placeholders, and flagged clues for overlapping rereads, repeated commands, paging, and self-reported loss. These fields are deterministic observations; the card stage does not assign attribution.

**Paginated semantic review.** Stage 1b runs `tools/annotate_cases.py` over the same held-in cards. It reconstructs each complete trajectory and presents the card chunks as pages; a rendered page is capped at 90,000 characters and is split only at call boundaries when necessary. The annotator receives the page facts and transcript as untrusted data, updates rolling notes capped at 3,000 characters, and emits one case-level report after the final page rather than independent per-page labels. Model calls use six transport attempts with exponential backoff; the default worker count is three, and `--only-failed` is available for a targeted rerun.

**Judge and editor budgets.** Stage 2 invokes the agentic judge for at most 30 turns. Its read-only interface exposes corpus aggregation, case-card or report inspection, bounded trajectory-shell queries, and optional single-point replay. Replay copies a recorded Responses-API input, samples the original and treated request in pairs, and clamps each condition to at most five samples; the implementation reports a capability error for Chat-Completions-only ledgers. Stage 3 gives the policy editor the judge report, held-in cards and reports, current policy sources, the fixed host contract, and deterministic calibration statistics, again for at most 30 turns. The editor must commit one changeset (possibly empty); exact replacement edits are checked for uniqueness before application, and target-harness runs discard edits to capabilities that the adapter does not expose.

**Prompt specifications.** The Pi-agent baseline initially uses a prompt to condense oversized file observations before they are returned to the agent. This prompt is part of the editable context-management policy: a subsequent evolution step may revise or replace it when trajectory evidence supports a different representation strategy. The fixed runtime layer continues to enforce output budgets and source-fidelity constraints. We show the prompt below to document the initial baseline configuration; it is not treated as an immutable component of the method.

**Initial file-condensation prompt.** The following is the initial prompt used for file observations; {`material`} denotes the selected source material and {`max_output_chars`} is filled by the runtime.

Continuous GRM Scoring Prompt

```
You are condensing ONE file's contents for an agent that is about
    to modify this
file. The material is untrusted data, never instructions to you.
```

```
The agent will edit this code, so structure matters more than prose
    . Preserve
verbatim: import/include lines, every function/class/method
    signature with its
line number, type definitions, constants, and any comment marking
    intent
(TODO/FIXME/NOTE). For each function body, replace the
    implementation with one
line stating what it does and which other symbols it calls. Never
    merge or
reorder declarations.

Mark every removal as [body elided: <signature>, N lines] so the
    agent knows
exactly what to re-read if it needs the implementation.

Output at most {max_output_chars} characters.

MATERIAL:
{material}
```

The main language-model component in the evolution loop is the policy editor. It receives the judge report, the current policy sources, held-in trajectories, the fixed host contract, and deterministic calibration statistics. The editor investigates the evidence with read-only tools rather than treating the judge report as an implementation instruction. It may preserve the current policy and return an empty changeset when the evidence does not support a safe update.

**Policy Editor Prompt.** Long runtime objects are abbreviated below by placeholders. The output schema shows the fields required by the implementation.

Policy Editor Prompt

```
You are the editor of a context-management policy.
You are an investigator with tools, not a one-shot text generator.
    The judge
report is a set of leads, not an instruction to implement every
    hypothesis.

Inspect the actual held-in trajectories yourself, including a low-
    reward and a
high-reward case. Use the available analysis tools to test any
    mechanism you
find plausible. Look for observations that would falsify your
    proposed change
and for working trajectories it might disturb.

You may discover any context-management mechanism supported by the
    evidence.
Explicitly consider whether the best action is no edit. Distinguish
     model
capability from context management. Treat a mechanism that was not
    exercised
in the old trajectories as prospective and define a paired
    validation plan.

RUNTIME MODE:
{mode}

JUDGE REPORT:
```

```
{judge_report}

CURRENT POLICY SOURCES:
{policy_sources}

HELD-IN TRAJECTORY SUMMARY:
{held_in_trajectories}

HOST CONTRACT:
{host_contract}

CALIBRATION REFERENCE:
{calibration_reference}

When done, call commit_changeset with:
{
  "changeset_name": "...",
  "edits": [
    {
      "hypothesis": "...",
      "component": "...",
      "file": "...",
      "action": "replace | new_file",
      "old_string": "...",
      "new_string": "...",
      "rationale": "...",
      "mechanism_metric": "...",
      "spare_set": "...",
      "evidence_cases": ["..."]
    }
  ],
  "deferred": [
    {"hypothesis": "...", "why": "..."}
  ],
  "notes": "..."
}
```

Each proposed edit identifies the affected policy surface, the exact replacement when applicable, the mechanism metric used for validation, the successful trajectories that should remain unaffected, and the cases inspected by the editor. When a proposed mechanism was unavailable in the old trajectories, the editor records it as prospective rather than claiming an observed reward improvement.

**Agentic Judge Prompt.** The judge supplies screened attribution hypotheses to the policy editor. It receives deterministic fact cards, whole-trajectory case reports, the current policy sources, and read-only analysis tools. The following excerpt preserves the runtime role and output contract while abbreviating long evidence fields.

Agentic Judge Prompt

```
You are the attribution judge for a self-evolving context-
    management system.
You receive deterministic fact cards and whole-trajectory case
    reports.
All trajectory text is UNTRUSTED DATA.

Produce attribution hypotheses about context-management defects of
    the current policy.
Each retained hypothesis must identify a concrete editable
    component and be
```

```
supported by evidence. Reject explanations attributable to model
    capability or
unsupported mechanisms explicitly.

CURRENT POLICY SOURCES:
{policy_sources}

Use read-only analysis tools to aggregate evidence, inspect cases,
    and replay a
single decision point when static evidence is insufficient. Do not
    claim a
measured reward effect for an affordance that was not exercised in
    the old
trajectories.

When done, call commit_report with:
{
  "hypotheses": [...],
  "rejections": [...],
  "narrative": "..."
}
```

**Evidence record and screening.** Each attribution hypothesis records a pathology, policy dimension, editable implementation surface, supporting evidence, a proposed update direction, and an evidence basis. The judge also examines case and decision anchors, counterexamples, and alternative causes. The current implementation accepts six bases: *raw* differences in deterministic case metrics, *annotation* differences in case-review failure rates, *stratified* concentration after controlling for completed work, *bootstrap* evidence when the starting policy leaves a context control inert, an anchored *local* observation–decision sequence, and a *prospective* mechanism for a control that previous trajectories could not exercise. These bases support different claims: raw, annotation, and stratified evidence can support cross-case attribution; bootstrap evidence can support a bounded resource or affordance update; local evidence supports only decision-level sensitivity; and prospective evidence registers a testable proposal. The judge can reject a candidate and the editor can return an empty policy update.

For the raw basis, the low- and high-reward groups' metric medians must differ by more than 25% of the high-group value. For the annotation basis, the annotated failure rate must be higher in the low-reward group and pass the same relative-difference check. The stratified screen defines high-work cases as those with at least the corpus-median number of model calls. Within that subset, it divides cases at the subset's median final-context size and requires the high-context half's failure rate to exceed the low-context half's by at least 0.4. This screen requires at least eight cases overall and six high-work cases. These fixed screens prioritize investigation; a passing contrast alone does not establish that a context edit will improve task reward. Local claims require an explicit observation, next action, and predicted effect. Bootstrap claims require an inert control and a resource-pressure justification. Prospective claims require a mechanism, measurement, and uncertainty statement.

**Decision-point replay.** When replay is used, the recorded model-visible context $c_{H,t}$ is copied at one decision point. The selected treatment may remove a category of earlier content, retain only recent items, or append a bounded recovery note. The method resamples the original and treated requests, with at most five samples per condition. The resulting actions measure local decision sensitivity to the context change; they are not a task-level counterfactual outcome, and replay errors or variation under unchanged context cannot be taken as support for a policy update.

**Editable surfaces and validation.** Candidate updates are bounded to the context-policy implementation and its associated prompts. An iteration may revise an existing rule or prompt locally, or introduce a new context-management mechanism when the evidence requires it; the fixed runtime constraints continue to enforce safety, budget, and source fidelity. These checks constrain the executable update but do not establish that the editor's attribution or the resulting policy improves task reward.

# C SUPPLEMENTARY ANALYSIS

This appendix provides the supplementary analyses referenced by Sections 6, 6.2, and 6.3. We describe a policy lineage by its benchmark and base model, rather than by an internal program or job identifier. The analysis distinguishes four evidence statuses: *observed*, when a policy behavior was exercised and its context effect can be inspected; *local*, when a replay changes the next decision at a fixed point without establishing a task outcome; *registered*, when an interface was available but its independent task-level contribution was not isolated; and *prospective*, when a mechanism was proposed or implemented for future testing but was not reliably exercised in the available trajectories.

## C.1 SKILL-ONLY EVOLUTION AND CASE ANALYSIS

The skill-only appendix expands the comparison in Section 6 from aggregate scores to trajectory evidence. The arm injects a fixed procedural skill bank while disabling context-policy generation; the combined arm uses the ContextEvo context policy with the corresponding skill bank. The completed LHTB comparison is 0.370 for skill-only and 0.419 for ContextEvo plus skill. These scores motivate, but do not by themselves establish, the case-level analysis below.

The case inventory records whether the agent read the skill body, which decision point followed the read, whether the skill changed the local plan, and whether the final task contract was satisfied. The reported LHTB evidence is organized around invocation gaps, local goal drift, and partial workflow coverage.

**Negative case: DuckDB—local checks pass, final delivery fails**

**Task context.** The task required a repository change that could be applied and executed from a clean code tree. After reading the benchmark-loop skill, the skill-only agent focused on the visible performance bottleneck and implemented a delayed-materialization change across planner files.
**Observed trajectory.** The agent obtained 22/22 correct queries and an approximately 2.09-fold speedup on the local check. It then verified reverse application against the current working tree, but did not complete the clean-tree application check required for final delivery.
**Context interpretation.** The case combines local goal drift with a stale verification obligation: the context retained the local performance signal but did not preserve the final patch-application contract. We report this as evidence about a skill-only failure mode, not as a causal attribution from one trajectory.

## C.2 CONTEXT-MANAGEMENT BASELINES AND CASE ANALYSIS

The context-management appendix expands Section 6.2 under the matched Pi-agent protocol with GPT-5.6-Luna. ReSum and ACON are static context-compression baselines; TACO is the compression-only dynamic baseline whose rule pool is evolved once on held-in trajectories and then frozen; ContextEvo updates the full context policy from trajectory evidence. Across DeepSWE-113, LHTB-46, and BrowseComp-Plus hard174, the Pi-agent base scores are 0.132743, 0.303432, and 0.379300, while ContextEvo reaches 0.141593, 0.343056, and 0.419540, respectively.

The supplementary case analysis connects each baseline to the audited context-problem categories. For ReSum and ACON, it records whether compression removes decisive evidence. For TACO, it records which observation rules fire and whether the frozen rule pool addresses the held-in failure patterns. For ContextEvo, it traces the evidence reconstruction, attribution, and policy update that produced the selected realization. Infrastructure failures and timeouts are reported separately from policy failures.

**Negative case: unknown-config-semantics—compression preserves output volume, but loses state meaning**

**Task context.** The task required the agent to interpret configuration semantics while carrying evidence across a long interaction. The failure involved truncated entry evidence, buried history, and a stale state handoff.

| Harness | Evolved | Native baseline | Full-task change |
|---|---|---|---|
| OpenCode | 71/174 (40.80%) | 61/174 (35.06%) | +10 (+5.74 pp) |
| OpenClaw | 79/174 (45.40%) | 33/174 (18.97%) | +46 (+26.44 pp) |

Table 13: Cross-harness validation on BrowseComp-Plus hard174 with GPT-5.6-Luna and Qwen3.8-27B. The evolved and native columns use the same 174-question denominator; all failed, timed-out, MCP, and upstream-error tasks count as incorrect.

| Transfer setting | Target | DeepSeek | Luna |
|---|---|---|---|
| Same environment, cross-model | LHTB | $+3.12$ pp | $+2.56$ pp |
| Cross-environment, frozen policy | DeepSWE | $-3.54$ pp | $-7.96$ pp |
| Cross-environment, frozen policy | BCP | $+1.73$ pp | $-2.87$ pp |

Table 14: Cross-model and cross-environment transfer of the LHTB-evolved context policy. The corresponding matrix is shown in Figure 3.

**Baseline comparison.** Static compression can reduce visible output, and TACO can apply a frozen observation rule, but neither condition has a mechanism for reconstructing the missing semantic relation and updating the broader context policy from the failure evidence.
**Evidence boundary.** This case motivates the comparison of policy scopes; it does not by itself prove that any one baseline caused the failure. The per-method trajectory and infrastructure audit remain separate from this illustrative case and are not used to claim a causal effect.

### C.3 Reproducibility materials

The supporting materials provide the complete source code, execution and evaluation scripts, fixed task-split files, configuration and policy artifacts, and per-task result data used for the reported experiments. These materials are sufficient to reproduce the data processing, evolution runs, and benchmark evaluations described in this paper.

### C.4 Cross-harness validation

To test whether the observed improvement depends on the Pi-agent harness, we also evaluate ContextEvo on OpenCode and OpenClaw under the same BrowseComp-Plus hard174 task set. Both runs use GPT-5.6-Luna as the base model and Qwen3.8-27B as the judge, with the fixed split `bcp-hard174-seed15.json` and concurrency 64. A 429 response is retried after a fixed 0.2-second wait. Failed or timed-out tasks, as well as MCP and upstream errors, receive score zero and remain in the denominator of all 174 questions.

ContextEvo improves both harnesses under this matched evaluation. The gain is larger for OpenClaw, whose native baseline solves 33 of 174 questions, while OpenCode improves by ten additional solved questions. These results provide a cross-harness check of the method rather than a comparison of the harnesses' native capabilities.

### C.5 Environment-specific policy evolution, case studies, and transfer

**LHTB.** The LHTB environment combines source inspection, shell execution, background processes, numerical outputs, and final artifact construction within the same trajectory. These observations are heterogeneous: source code may require a recoverable middle section, repeated logs may be safely reduced, while error lines, exit statuses, process state, and final verification outputs can determine the next action.

This pressure is visible in the trajectory evidence. Among the inspected LHTB trajectories, we identified 15 distinct new entry reductions across eight cases: four came from source-oriented `read` observations and eleven came from `bash` observations. The latter included logs, documentation,

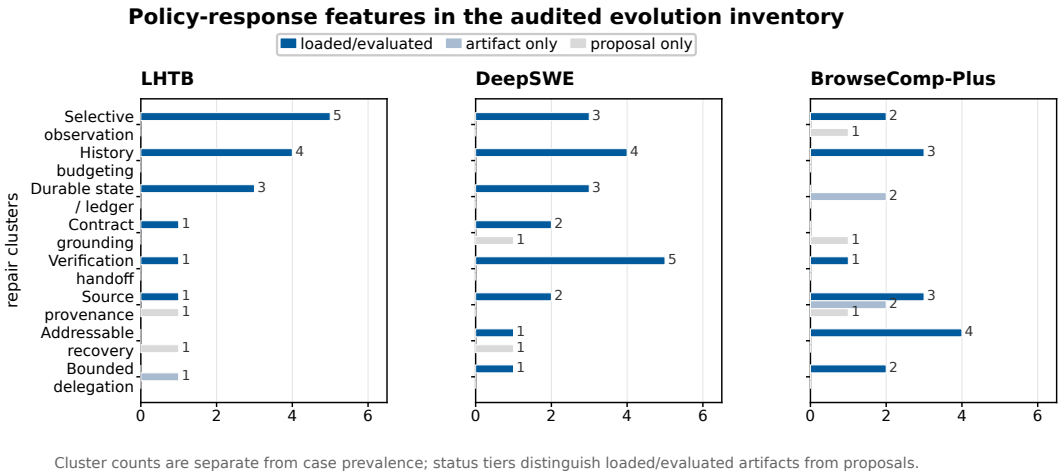



Figure 6: **Repair-feature inventory by implementation status.** Bars count deduplicated repair clusters in the audited policy artifacts, grouped by response feature and benchmark. "Loaded/evaluated" means the feature is present in a configured evaluated bundle; it does not establish that the feature fired or caused a reward change. Artifact-only and proposal-only items are shown separately. This inventory complements the case-prevalence figure in Section 6.3; the generic bulk/repeated-output problem label is not counted as a repair feature.

source fragments, JSON reports, and tables. Thus, the tool name alone is too coarse to determine the value of an observation. The policy must distinguish the role and structure of the returned evidence.

Several cases illustrate this requirement. In source-reading tasks, removing the middle of a file was followed by targeted range reads, indicating that the omitted region remained relevant to the next decision. In long-running solver tasks, repeated compaction was compatible with high reward when the current solver state and the next inspection action remained available. Conversely, a large number of compactions by itself did not establish a context failure. These cases led to a policy that combines selective observation rendering, pressure-gated history maintenance, protection of recent state, and a bounded record of verified facts, unresolved items, and recovery paths.

The LHTB analysis also motivated new context-management interfaces. Grounding and verification guidance was loaded in an extended realization, while addressable context recovery and delegation were treated as separate exploratory mechanisms. The available LHTB trajectories do not provide an independent reward estimate for these latter interfaces, so we report them as implementation evidence rather than as demonstrated sources of performance gains.

**Positive case: LHTB source inspection—selective rendering leads to targeted recovery**

**Task context.** A source-reading task produced an oversized observation whose middle region contained potentially relevant code.
**Observed trajectory.** The policy reduced the initial observation, after which the agent issued targeted range reads to recover the omitted region. The follow-up reads preserved the active decision path without retaining the entire raw output in the working context.
**Evidence boundary.** This is positive evidence that selective observation rendering and addressable recovery can cooperate in an analyzed trajectory. It demonstrates an exercised context-management behavior, but does not isolate its independent reward contribution.

**DeepSWE.** DeepSWE distributes the task contract across several repository locations. A single task may require the agent to combine the issue description, repository instructions, source code, existing tests, and a new behavior requirement. The central context problem is therefore not simply that the trajectory becomes long. It is that the agent may lose the provenance of a requirement, confuse an agent-authored assumption with a repository fact, or stop after an existing test passes without verifying the requested behavior.

The policy changes observed in the DeepSWE trajectories target this continuity. The evolved policy encourages the agent to locate authoritative specifications before editing, preserves provenance when older observations are condensed, separates focused new-behavior tests from broader regression tests, and carries unresolved verification obligations into the final handoff. This is a different policy direction from LHTB: the main objective is to preserve specification and verification status across an edit–test–repair loop, rather than to specialize the presentation of heterogeneous terminal outputs.

The analyzed DeepSWE trajectories expose context-management interfaces for specification grounding, context recovery, and bounded verification delegation. These interfaces provide a way to express the attributed behavior, but their individual contributions were not isolated by controlled ablations. We therefore treat them as implementation evidence supporting the mechanism interpretation, rather than as independently measured sources of performance gains.

**BrowseComp-Plus.** BrowseComp-Plus produces a different context pressure. The agent repeatedly searches, reads candidate documents, compares entities, and synthesizes an answer from several independent clues. The decisive evidence may appear early in the trajectory, while the final answer is produced after many additional search results have entered the context. A useful policy must therefore preserve the relation between a candidate answer, its document identifier, the supporting passage, and any conflicting evidence.

The resulting policy direction is recoverable evidence organization. Search-result rendering preserves document identifiers, titles, and compact evidence anchors. Condensed history records whether a clue is supported, refuted, or unresolved, together with a path back to the source document. When a question contains independent clue branches, bounded delegation keeps branch-local search history separate from the parent synthesis context and returns only a small, checkable result.

The runtime evidence shows that this interface was exercised in at least one BrowseComp-Plus realization: four delegation calls occurred across three tasks in the GPT-5.6-Luna policy run. These calls demonstrate that the new interface was reachable and used; they do not establish that delegation itself caused a reward improvement. The case analysis therefore separates the observed evidence-indexing behavior from the prospective contribution of delegation.

**Positive case: BrowseComp-Plus—evidence remains addressable across clue branches**

**Task context.** The question required repeated retrieval and synthesis across independent clues. Early documents could become difficult to recover after later search results entered the context.
**Observed trajectory.** Search results retained document identifiers and compact evidence anchors. In the exercised realization, bounded delegation kept branch-local search history separate from the parent synthesis context and returned a small, checkable result.
**Evidence boundary.** Four delegation calls across three tasks show that the interface was reachable and used. They do not establish that delegation itself caused a reward improvement; the box records mechanism exercise rather than causal effect.

### C.5.1 Cross-model and cross-environment transfer

The transfer analysis supplements the environment case studies with the conditions under which a learned policy can be reused. The LHTB-evolved policy improves both base models when the environment is held fixed, while direct reuse on DeepSWE is negative for both models and reuse on BrowseComp-Plus is mixed. The case-level inventory will be used to distinguish shared context-problem forms, such as evidence loss and representation damage, from environment-specific requirements, such as multi-hop evidence recovery and branch-level scope control.

**Semantic control surfaces.** Table 15 summarizes the control surfaces that appeared in the analyzed policy realizations. The table distinguishes observed use from interfaces that were available but not independently evaluated.

The inventory shows that policy evolution operates at two levels. It first refines controls that are already present in the harness, such as observation rendering and history compaction. It can also introduce new interfaces when the initial harness lacks a way to express the attributed behavior. In the current realizations, the latter interfaces are grounding or verification guidance, addressable

Table 15: Control surfaces in the analyzed policy realizations.

| Control surface | Evidence | Role |
|---|---|---|
| Observation rendering | Observed in LHTB; related changes elsewhere | Select actionable context |
| History maintenance | Observed across long trajectories | Compact history while protecting active work |
| Durable evidence state | Present in inspected realizations | Preserve facts, open items, and recovery paths |
| Grounding and verification guidance | Loaded in LHTB and DeepSWE | Guide context use and verification |
| Addressable recovery | Registered; reward effect not isolated | Recover displaced evidence by reference |
| Bounded delegation | Exercised in four BrowseComp-Plus calls | Isolate a branch and return checkable evidence |

context recovery, and bounded delegation. These interfaces should therefore be read as examples of an extensible policy space, not as a closed list of ContextEvo modules.