HARP: ON-POLICY DISTILLATION OF HIERARCHICALLY ROUTED PLANNING, EXECUTION, AND REVISION EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tool-use agents remain difficult to train because task-level outcome rewards provide sparse credit across trajectories involving many interdependent decisions. Process-level supervision alleviates this sparsity with finer-grained feedback, but typically treats intermediate errors through a largely homogeneous signal. In interactive environments, however, failures are structurally heterogeneous: an agent may pursue an incorrect subgoal, execute a valid plan with an incorrect tool call, or fail to diagnose and recover from an earlier mistake. Effective supervision should therefore capture not only where credit should be assigned, but also what kind of failure occurred and how it should be corrected. We introduce *HARP* (Hierarchically Arranged Role Policies), a training framework that decomposes failures into planning, execution, and recovery errors and assigns them to role-specialized Planner, Executor, and Repairer components coordinated by verification and routing mechanisms. HARP uses this structured system as a training-time teacher and distills its role-specific corrective supervision on-policy into a single base model, avoiding the full scaffold at deployment. On AppWorld with Qwen3-1.7B, 4B, and 8B, the distilled HARP model improves task goal completion over the base model at every size and outperforms GRPO at 1.7B and 4B, while GRPO remains stronger at 8B. Component ablations and routing diagnostics further characterize the contribution of each role.

1 INTRODUCTION

Recent frontier models (Xiaomi LLM-Core Team, 2026; Team et al., 2026; DeepSeek-AI, 2026) illustrate a powerful post-training strategy: develop specialized teachers and consolidate their expertise through multi-teacher on-policy distillation (MOPD). These systems organize teacher specialization around task domains and reasoning-effort levels, allowing a single student to learn from complementary experts on its own rollouts. This raises a different question for long-horizon tool use: when the task domain remains fixed but the expertise required changes over the course of a trajectory, how should teacher specialization be organized?

This question is especially important in long-horizon agentic reinforcement learning, where many interdependent decisions share a delayed terminal outcome (Chen et al., 2025). A failed rollout reveals neither which decision caused the failure nor whether the model should revise its plan, improve its tool execution, or recover from an earlier mistake. In group-relative policy optimization (GRPO) (Shao et al., 2024), a rollout group in which every trajectory receives the same outcome produces zero or nearly zero relative advantages, providing little guidance about which earlier decision should change (Feng et al., 2025). Process-level methods reduce this sparsity by redistributing terminal rewards across steps or deriving intermediate scores from trajectory comparisons and verifiers (Feng et al., 2025; Li et al., 2026). Although these methods improve the temporal resolution of credit, they often aggregate evidence about planning, execution, and recovery into the same reward or advantage. The central bottleneck is therefore not only where to assign credit, but also which capability the resulting update should improve.

Intuitively, completing a complex, long-horizon task can benefit from a hierarchical division of labor. Consider launching a product: senior leadership sets the strategic direction, team leads translate

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.

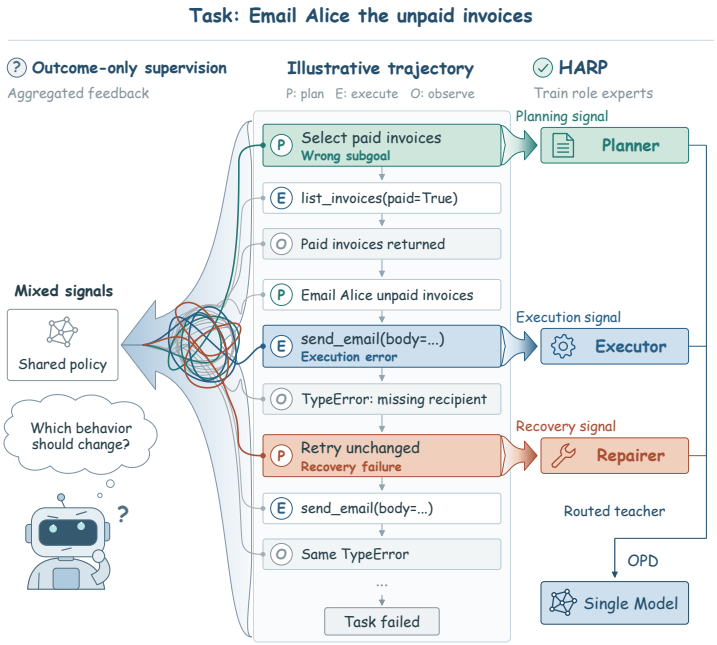


Figure 1: Overview of HARP. Compared to outcome-only supervision, which mixes planning, execution, and recovery failures into one signal, HARP routes role-specific supervision to specialized teachers and then distills them into one model.

broad objectives into concrete implementation plans, engineers execute those plans, and testing and review teams assess the results and identify areas for improvement. These roles contribute complementary expertise at different levels of abstraction and with distinct functional responsibilities, all toward a shared objective. A similar perspective can inform the design and training of long-horizon agents: high-level planning determines what to pursue, lower-level planning translates objectives into actionable steps, execution turns those steps into concrete actions, and verification and recovery help detect problems and restore progress.

Our analysis of long-horizon training trajectories reveals three recurring modes of failure. First, an agent may choose an incorrect or incomplete subgoal, resulting in a planning failure. Second, it may pursue a reasonable subgoal but fail to carry it out, for example by selecting the wrong API or supplying incorrect arguments, resulting in an execution failure. Third, it may fail to diagnose and recover from an earlier mistake, resulting in a repair failure. Examples of these failure modes are shown in Appendix A. Independent work on web agents similarly separates high-level planning, low-level execution, and re-planning, and finds low-level execution to be the dominant bottleneck (Aghzal et al., 2026). Reflexion further shows that explicit feedback after failure can improve subsequent attempts (Shinn et al., 2023). These failures require different corrections, yet terminal outcomes and homogeneous process signals do not indicate which responsibility caused the trajectory to fail.

This finding motivates organizing teacher expertise not only around what domains teachers know, but also around what responsibilities they fulfill. We introduce *HARP* (Hierarchically Arranged Role Policies), which trains three role-specialized experts and consolidates their complementary behavior into a single deployable model. The Planner proposes subgoals, the Executor carries out a selected subgoal through grounded tool calls, and the Repairer diagnoses deviations and selects corrective actions. During interaction, verifiers check whether the current plan and tool execution advance the task and provide evidence to a router that selects the relevant expert. The resulting routed system serves as a training-time teacher, as illustrated in Figure 1. Through on-policy distillation, the student receives role-specific supervision on the states that it visits itself, reducing the mismatch between training examples and deployment behavior (Agarwal et al., 2024). At deployment, the distilled student runs alone without the expert harness.

Our contributions are threefold. First, we identify the problem of role-entangled supervision in long-horizon tool-use agents and provide a theoretical analysis of why separating planning, execution, and recovery feedback can preserve information that is lost by a single aggregated signal. Second, we present HARP, which trains Planner, Executor, and Repairer experts, combines them with evidence-based verification and routing, and transfers their behavior into a single model through on-policy distillation. Third, we establish an AppWorld evaluation across Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, using terminal task goal completion (TGC) as the primary measure of task performance and comparing HARP with the Base model and GRPO, alongside component ablations and routing/progress diagnostics. Together, these contributions connect a theoretical account of structured feedback with a practical training framework and an evaluation of long-horizon task completion.

2 Related Work

Long-Horizon Tool-Use Agents. Long-horizon tool-use agents must coordinate reasoning, API calls, and environment feedback over a sequence in which early mistakes can affect all later decisions. AppWorld makes this setting explicit by evaluating whether an agent completes multi-application tasks in a stateful world by checking the resulting environment state (Trivedi et al., 2024); related work also treats such interactions as long-horizon decision making under partial observability (Chen et al., 2025). ReAct interleaves reasoning with actions (Yao et al., 2023). The remaining challenge is to determine which capability a failed interaction reveals as deficient. HARP addresses this challenge by making the source of supervision explicit.

Hierarchical Planning and Execution. One response to long-horizon difficulty is to structure a task into higher-level plans or subgoals and lower-level tool execution. Sub-goal Distillation transfers high-level subgoals to a smaller language agent and trains planning and execution behavior separately (Hashemzadeh et al., 2025); Plan-and-Act similarly separates a Planner that produces structured plans from an Executor that carries them out (Erdogan et al., 2025); and HiPER factorizes the policy into planner and executor levels with hierarchical advantage estimation (Peng et al., 2026). AdaPlanner and Reflexion show that feedback can further support plan revision or a better subsequent attempt (Sun et al., 2023; Shinn et al., 2023). HARP builds on this line by treating recovery after an observed deviation as a separately trainable Repairer responsibility, then using verifiers and a router to compose the three roles before distilling them into one deployable model.

Step- and Process-Level Credit Assignment. Process supervision replaces a single outcome label with judgments about intermediate steps and has shown the value of finer-grained feedback in multi-step reasoning (Lightman et al., 2024). Agent-training methods extend this idea to interactive trajectories: GiGPO compares grouped states across rollouts to obtain step-level advantages, HCAPO uses hindsight reasoning to refine value estimates, BEACON estimates credit at multiple temporal scales, and VICT uses verifiers to trace credit through long interactions (Feng et al., 2025; Tan et al., 2026; Wang et al., 2026; Li et al., 2026). These methods improve the answer to *when* a consequential decision occurred, but a locally correct action need not advance the task and a learned process signal can become a proxy for the terminal objective. An executed-replay audit likewise found that LLM-judge (correctness) scores, model confidence, and outcome-conditioned likelihood do not reliably identify a step's causal effect on final success (Zhang, 2026). HARP complements temporal credit assignment by conditioning supervision on whether the deficiency concerns planning, execution, or recovery.

On-Policy Distillation and Structured Teacher Signals. On-policy distillation (OPD), including on-policy self-distillation (OPSD), trains a student on states that it generates itself while using a teacher or privileged context to provide token- or step-level guidance, reducing the mismatch caused by learning only from teacher trajectories (Agarwal et al., 2024). Recent agent-oriented methods such as OPID extract episode-level and critical-step skills from completed on-policy trajectories and route the relevant skill during training (Yang et al., 2026). Multi-teacher distillation has progressed from offline fusion of heterogeneous model distributions and structured multi-agent interactions (Wan et al., 2024; Chen et al., 2024) to MOPD, which independently trains domain-specialized RL teachers and routes each student-generated trajectory to the teacher associated with its prompt domain before consolidating their capabilities into one policy (Ma et al., 2026). Work on model-harness co-evolution likewise uses a richer training system to generate trajectories that improve a model while keeping the deployed system simpler (Chen et al., 2026). HARP combines these ideas

but uses a different routing variable: MOPD asks which domain a prompt belongs to, whereas HARP asks which corrective capability is needed at the current interaction state. Teacher assignment can therefore change within a trajectory as evidence of planning, execution, or recovery failures emerges.

3 Theoretical Analysis

3.1 Information loss under aggregated supervision

Let $C = (X, H_t)$ contain the task and the history shared by both methods, and let U be a finite-valued corrective intervention. We compare structured training feedback Z with a matched aggregation $R = g(Z, C)$ on the same joint distribution. Here Z can carry role identity and an expert target in addition to scalar scores. Define the discarded information as $\mathcal{G} = I(U; Z \mid C, R)$. The explicit *non-sufficiency assumption* $\mathcal{G} > 0$ means that aggregation removes information about U beyond the shared context.

Proposition 1 (aggregation loss). For optimal prediction under log loss, let $\mathcal{L}_F^\star = \inf_\nu \mathbb{E}[-\log \nu(U \mid C, F)]$, where ν ranges over all conditional distributions on the intervention alphabet. Then

$$\mathcal{L}_R^\star - \mathcal{L}_Z^\star = H(U \mid C, R) - H(U \mid C, Z) = \mathcal{G} \geq 0. \tag{1}$$

Thus aggregation necessarily incurs a positive log-loss gap when $\mathcal{G} > 0$. For any fixed bounded decision loss d, define $\mathcal{B}(F) = \inf_\delta \mathbb{E}[d(U, \delta(C, F))]$. Retaining Z can simulate every decision based on R, so

$$\mathcal{B}(Z) \leq \mathcal{B}(R). \tag{2}$$

This is a standard comparison of attainable Bayes risks (Cover & Thomas, 2006; Blackwell, 1953); whether the inequality is strict depends on the decision loss. Proofs are in Appendix B.1.

3.2 Trajectory success and OPD retention

We separately consider an idealized renewal model with unlimited *inference* budget. A task requires N progress stages at a fixed subgoal granularity ($k = 1/N$). At each stage, a new local plan is feasible with probability p, and a feasible execution succeeds with probability $e > 0$; an infeasible initial plan makes no progress. Following either failure, a recovery episode reaches a feasible retry with probability r, and otherwise makes no further progress. Episodes may include multiple repair outputs. Assume stationary probabilities and independent stages and retries.

Let q be success starting from a feasible attempt and s success starting from a new local plan. Then

$$q = e + (1 - e)rq = \frac{e}{1 - (1 - e)r}, \qquad s = pe + (1 - pe)rq = \frac{e[p + (1 - p)r]}{1 - (1 - e)r}. \tag{3}$$

Hence $J_\infty(p, e, r, N) = s(p, e, r)^N$, allowing an initially infeasible plan to be repaired. Under a fixed training budget, let primed parameters describe the final OPD *student* and tilded parameters the directly mixed student. For positive success probabilities,

$$J_{\mathrm{OPD},\infty} > J_{\mathrm{mix},\infty} \quad \Longleftrightarrow \quad N' \log s(p', e', r') - \widetilde{N} \log s(\widetilde{p}, \widetilde{e}, \widetilde{r}) > 0. \tag{4}$$

Componentwise improvements in p, e, r together with a nonincreasing N guarantee weak dominance; whether specialization delivers such improvements is tested empirically in Section 5.

OPD retention admits a separate finite-horizon bound. For a teacher and student with the same observable history, action interface, environment, and horizon H, let ϵ be their average action-distribution total variation on the *student's* visited histories. With terminal success in $[0, 1]$,

$$J_H(\pi_{\mathrm{student}}) \geq J_H(\pi_{\mathrm{teacher}}) - H\epsilon. \tag{5}$$

Hence any teacher advantage over mixed training larger than $H\epsilon$ survives distillation. Appendix B.2 extends the success model to finite interaction budgets and proves Eq. 5.

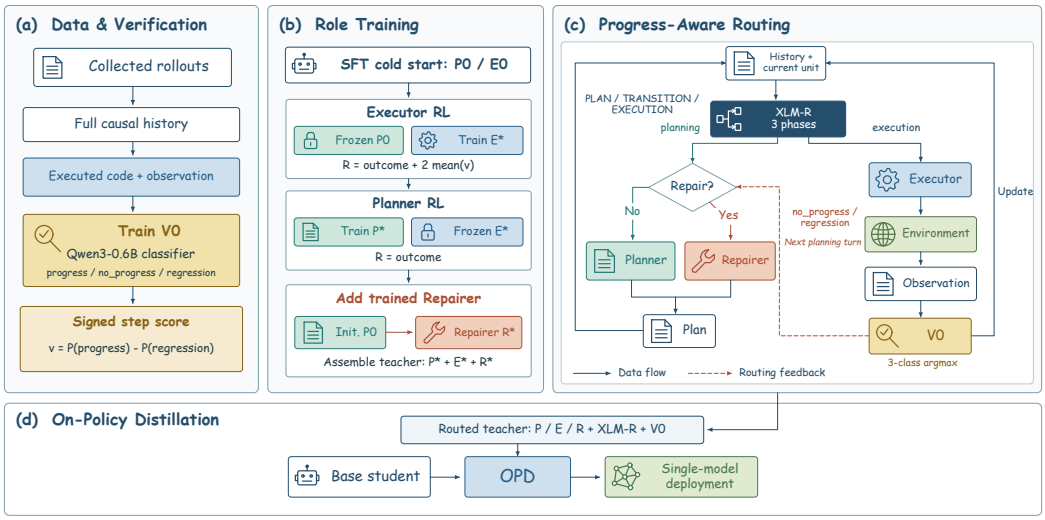


Figure 2: HARP's training and routing modules. (a) Observed transitions train the three-class progress verifier V_0. (b) SFT cold-starts Planner and Executor; RL first updates Executor with Planner frozen, then updates Planner with the trained Executor frozen. A trained Repairer is added to the expert system. (c) XLM-R classifies planning, transition, and execution units; V_0 selects Planner or Repairer after an observation. (d) The assembled expert teacher provides on-policy supervision to a student initialized from the original base model.

4 Methodology

4.1 Overview

HARP converts a structured training harness into a single deployable policy. Given a task X and interaction history H_t, the Planner π_{P} proposes a next subgoal, the Executor π_{E} generates grounded tool actions, and the Repairer π_{R} proposes a revised plan after an observed deviation. A phase classifier controls handoffs between planning and execution; a separate progress verifier decides whether the next planning turn requires repair. Accepted outputs and observations remain in the shared textual history across handoffs, subject to each model's context limit.

Figure 2 summarizes the training sequence. SFT first cold-starts the Planner and Executor. We then freeze the SFT-initialized Planner and train the Executor by RL. Next, we freeze the resulting trained Executor and train the Planner by RL. We add the trained Repairer and assemble these three experts with the phase classifier and progress verifier into a routed teacher. Finally, OPD transfers this fully assembled teacher's supervision to a separate student initialized from the original base model. Appendix C gives the stage-specific implementation details.

4.2 Rollout Collection and Progress Verification

A plan–execution turn produces a plan G_t, an actually executed tool action A_t, and its paired observation O_{t+1}. We retain the causal context $F_t = (X, H_t, G_t, A_t, O_{t+1})$ for verification and correction, which contains only information available up to the current observation.

We fine-tune Qwen3-0.6B (Yang et al., 2025) as a three-class sequence classifier on 9,622 labeled transitions selected from failure points in two-role Planner–Executor rollouts. The frozen verifier V_0 maps F_t to a distribution $\mathbf{v}_t$ over $\mathcal{C} = \{\texttt{progress}, \texttt{no_progress}, \texttt{regression}\}$, with predicted label $\widehat{z}_t = \arg\max_c v_t(c)$. Labels reflect whether the observation shows progress on the task, not merely whether the call executed. A task-disjoint heldout set contains 1,001 examples. Label construction, exclusions, class balance, and full-history training settings are in Appendix C.2.

In the Executor RL stage, the trajectory reward is

$$R_{\mathsf{E}}(\tau) = R_{\text{out}}(\tau) + 2\,\overline{V}(\tau), \qquad \overline{V}(\tau) = \frac{1}{|\mathcal{S}(\tau)|} \sum_{s \in \mathcal{S}(\tau)} s, \tag{6}$$

where $\mathcal{S}(\tau)$ is the list of effective step scores. For an ordinary executed action with a paired observation, $s_t = v_t(\texttt{progress}) - v_t(\texttt{regression})$; protocol failures contribute a fixed penalty as detailed in Appendix C.3. The progress term thus acts as a trajectory-level bonus that can distinguish rollouts with the same terminal outcome. The subsequent Planner RL stage uses $R_{\mathsf{P}} = R_{\text{out}}$.

4.3 Sequential Reinforcement Learning of Planner and Executor

SFT produces the initial role policies π^0_{P} and π^0_{E}, which we then optimize with GRPO in sequence. First, with the Planner fixed at π^0_{P}, the Executor is trained from π^0_{E} to maximize R_{E}, yielding π^*_{E}. We then freeze π^*_{E} and train the Planner from π^0_{P} to maximize R_{out}, yielding π^*_{P}. Both stages sample complete interactions, but gradient masks retain only the active role's generated tokens; the partner policy, router, phase classifier, and verifier remain fixed. The resulting pair $(\pi^*_{\mathsf{P}}, \pi^*_{\mathsf{E}})$ is retained for teacher assembly.

4.4 Learning a Repairer from Failed Rollouts

We collect failed or stagnant contexts alongside normal controls and compare alternative next plans by replaying short branches from the same environment state with a frozen Executor. Verifier scores prefilter candidates; actual replay and Codex-assisted semantic review check whether the proposed plan repairs the observed problem and respects the task constraints. Each training example pairs the context up to the error observation with the first accepted corrective plan.

The strict correction set contains 17 repair examples from 13 tasks, mixed with 34 ordinary planning examples. The Repairer is initialized from the SFT-trained Planner π^0_{P} of the corresponding model size and fitted by LoRA SFT. At interaction time it proposes a revised plan conditioned on F_t, which the trained Executor then carries out. We add the resulting Repairer π^*_{R} to $(\pi^*_{\mathsf{P}}, \pi^*_{\mathsf{E}})$ and freeze all three experts for OPD. Appendix C.4 gives the 750-state candidate collection and filtering procedure. The Repairer's contribution to terminal success is evaluated by the component ablation.

4.5 Progress-Aware Routing

The phase classifier C_ϕ is XLM-R-base (Conneau et al., 2020), trained to label each completed sentence or code unit, given its preceding context, as PLAN, TRANSITION, or EXECUTION. The stateful router retains accepted token prefixes across model handoffs. Planner or Repairer continues on a planning unit; a transition unit is accepted before handing off to the Executor. If the Planner begins emitting execution content, control passes to the Executor. We refer to this two-role Planner–Executor router as Router2 and to its progress-aware extension with the Repairer as Router3.

After each action and its observation, Router3 assigns the next planning turn to the Planner if $\widehat{z}_t = \texttt{progress}$ and to the Repairer otherwise. The first turn starts with the Planner. Following a Repairer-to-Executor handoff, the rest of that assistant turn remains with the Executor; the next observation is assessed anew. Ordinary Planner turns retain the Router2 re-planning logic. Invalid verifier outputs, overlength inputs, or inference failures are logged and fall back to the Planner. Further routing settings are in Appendix C.5.

4.6 On-Policy Distillation into a Single Model

We adapt the SOD implementation (Zhong et al., 2026) for routed teacher scoring. The frozen teacher comprises π^*_{P}, π^*_{E}, and π^*_{R}; each Qwen3 student is initialized from its original base model and generates its own multi-turn trajectories. The assembled teacher scores the sampled assistant tokens under their observed prefixes, with role assignment provided by the phase classifier and progress-aware routing.

Let ℓ^S_{ij} be the rollout student's log-probability of token j in trajectory i, and let ℓ^T_{ij} be the selected teacher's log-probability of that same token. The implemented update augments the GRPO advan-

tage with routed feedback:

$$A_{ij}^{\text{total}} = A_{ij}^{\text{GRPO}} + \lambda w_{i,k(j)}(\ell_{ij}^{T} - \ell_{ij}^{S}), \qquad \lambda = 1. \tag{7}$$

Here $k(j)$ identifies a contiguous assistant-token segment, and w_{ik} is a disagreement-based segment weight defined in Appendix C.6. Tool observations and padding are masked out. Teacher scores and weights are computed after sampling and held fixed for the policy update. At deployment, only the student remains and uses the standard single-model environment interface.

5 Experiments

5.1 Experimental Setup

Benchmark and splits. We evaluate on AppWorld (Trivedi et al., 2024), where agents complete multi-application tasks through API calls and task completion is checked against the resulting environment state. AppWorld provides 90 training tasks, 57 development tasks, Test-N (168 tasks), and Test-C (417 tasks). We report task goal completion (TGC) on the two official test splits. Stage-specific subsets and final-model selection details are given in Appendix C.7. A task counts as complete only when all of its required state-based tests pass. All methods are evaluated under the same AppWorld interaction protocol from AgentEvolver (Zhai et al., 2025), with greedy decoding and a limit of 30 environment interactions per rollout.

Models. We study Qwen3-1.7B, Qwen3-4B, and Qwen3-8B from the Qwen3 family (Yang et al., 2025). The Base policies and OPD students use these three sizes. For each student size, the Planner, Executor, and Repairer teachers use the matching model size; the XLM-R phase classifier and Qwen3-0.6B progress verifier are shared auxiliary modules. *Base* denotes the corresponding model without our additional training, and each student is initialized from this original base model. HARP in the main comparison denotes the final distilled single model.

Baselines. We use two baselines: the unmodified Base model and GRPO (Shao et al., 2024) trained using AgentEvolver's GRPO implementation and AppWorld interaction protocol (Zhai et al., 2025). We label the latter *GRPO (AgentEvolver)* to identify its implementation source. It is trained from the corresponding base model on the AppWorld training tasks for 60 updates, each using 32 tasks with 8 rollouts per task. Appendix C.3 summarizes its settings.

Implementation and reporting. Appendix C gives data construction, optimization settings, and evaluation accounting for each training stage. All TGC values are from a single greedy-decoding run per configuration.

5.2 Main Results

We compare HARP with the unmodified Base model and GRPO trained using AgentEvolver's implementation and interaction protocol. Table 1 presents the main results on AppWorld Test-N and Test-C across Qwen3-1.7B, Qwen3-4B, and Qwen3-8B. HARP is evaluated as a standalone distilled model.

HARP achieves higher TGC than Base at all three model sizes on both splits. Its advantage over GRPO is largest for the smaller models. At 1.7B, HARP reaches 13.1% on Test-N and 3.8% on Test-C, exceeding GRPO by 12.5 and 3.8 percentage points. At 4B, HARP reaches 19.6% and 8.4%, with gains of 6.5 and 3.1 points over GRPO. These results suggest that the routed-teacher training approach is particularly promising for improving smaller standalone agents.

The advantage is not uniform across model sizes. At 8B, GRPO achieves 36.3% on Test-N and 12.0% on Test-C, compared with HARP's 32.7% and 8.4%; HARP trails by approximately 3.6 points on each split. Increasing HARP from 4B to 8B improves Test-N performance, while Test-C remains at 8.4%. HARP's advantage over GRPO is therefore most pronounced at smaller scales; Section 6 discusses a possible explanation.

Table 1: Main AppWorld results, measured by TGC (%). Within each model size and test split, **best** scores use darker muted-yellow shading, and second-best scores use lighter yellow shading; ties share the same marking. Test-N and Test-C contain 168 and 417 tasks, respectively.

Model	Method	Test-N TGC	Test-C TGC
Qwen3-1.7B	Base	0.0	0.0
	GRPO (AgentEvolver)	0.6	0.0
	HARP	**13.1**	**3.8**
Qwen3-4B	Base	5.4	2.4
	GRPO (AgentEvolver)	13.1	5.3
	HARP	**19.6**	**8.4**
Qwen3-8B	Base	17.9	6.7
	GRPO (AgentEvolver)	**36.3**	**12.0**
	HARP	32.7	8.4

Table 2: Component comparisons across three Qwen3 sizes (TGC, %). Base and OPD repeat Table 1. The pair without/with Repairer uses Router2/Router3, respectively.

	Qwen3-1.7B		Qwen3-4B		Qwen3-8B	
Configuration	Test-N	Test-C	Test-N	Test-C	Test-N	Test-C
Base (single model)	0.0	0.0	5.4	2.4	17.9	6.7
Base Planner + trained Executor	1.8	0.0	13.1	3.6	20.8	7.7
Trained Planner + base Executor	0.6	1.0	10.7	3.8	15.5	4.3
Trained Planner + Executor	10.7	3.4	20.2	6.5	32.7	7.9
Trained Planner + Executor + Repairer	10.7	2.9	27.4	7.0	32.7	7.7
OPD student (HARP)	13.1	3.8	19.6	8.4	32.7	8.4

5.3 Component Ablations

We compare six configurations at each model size to examine role composition and consolidation (Table 2). Base and the OPD student reuse the corresponding main results. In the routed configurations, size-matched trained policies are assigned to one or both role interfaces. Router3, used with the Repairer, prevents further re-planning after a Repairer-to-Executor handoff within the same turn.

The single-role replacements favor the trained Executor on Test-N, while their Test-C ordering varies by size. Adding the Repairer has its largest observed benefit at 4B: Test-N increases from 20.2% to 27.4% (34 to 46 successful tasks), and Test-C from 6.5% to 7.0%. At 1.7B and 8B, Test-N is unchanged and Test-C decreases slightly. The OPD student also does not uniformly match the routed system: at 4B, its Test-N score is 19.6% versus 27.4% with the Repairer, while Test-C rises from 7.0% to 8.4%. The Repairer thus helps most at 4B, and at that size distillation trades some Test-N performance for higher Test-C accuracy.

5.4 Routing and Verifier Diagnostics

The XLM-R phase classifier C_ϕ correctly labels 5,642 of 5,667 strict heldout sentence/code units (**99.56%**) as planning, transition, or execution (confusion matrix in Appendix C.5). The progress verifier V_0 reaches **97.90%** accuracy ($980/1001$), macro-F1 0.8169, and balanced accuracy 0.7659 on its task-disjoint heldout set, but detects only $3/9$ regression examples, all from one task. In a separate Qwen3-8B routing diagnostic, adding the Repairer lowers the fraction of post-action observations that the same frozen V_0 labels `no_progress` from 43.39% to 39.24% on Test-N and from 54.88% to 45.61% on Test-C. Because the two configurations visit different states, this comparison is descriptive; Appendix C.7 gives the metric definition and exact counts.

6 Discussion and Limitations

HARP improves over Base at all three sizes, but outperforms GRPO only at 1.7B and 4B. One hypothesis is that weaker policies more often produce rollout groups in which every trajectory receives zero terminal reward, leaving GRPO's outcome-based term without reward contrast (Shao et al., 2024). HARP may bypass this bottleneck: its Executor reward adds transition-level verifier scores that can separate rollouts with identical outcomes, and the improved execution behavior, reused throughout a trajectory, makes success more reachable for subsequent Planner training. The assembled experts then supervise the OPD student on its own histories.

Several limitations constrain this interpretation. Each TGC value comes from a single evaluation run, without estimates of seed variation. Repairer gains vary across sizes and splits, and at 4B on Test-N the OPD student falls below the routed system. A matched-cost comparison would need to account for expert training, verifier construction, repair-data collection, and OPD. Because the same verifier supplies training feedback and diagnostics, its predictions require validation against independent task outcomes. Finally, the theoretical results are conditional; they do not show that the implemented procedure attains Bayes-optimal risks, stationary renewal behavior, or the required distillation error bound.

AppWorld tasks are bounded, with explicit objectives and state-based evaluation (Trivedi et al., 2024), so the three roles in HARP should be viewed as one useful decomposition for this setting. Deeper and task-dependent hierarchies, continuous representations of role requirements, and training of higher-level coordinators are left to future work (Appendix D).

7 Conclusion

We presented HARP, which trains planning, execution, and repair experts, composes them through progress-aware routing, and distills them on-policy into a single model. Our analysis shows that aggregating role-specific feedback into one signal can discard information about the needed correction, and gives conditions under which a teacher's advantage survives distillation. On AppWorld, HARP improves over the base model at all three sizes and over GRPO at 1.7B and 4B, with the largest gains for the smallest model. These results suggest that organizing teachers around responsibilities, in addition to domains, is a promising direction for training compact long-horizon agents.

AI Use Statement

We used generative AI tools for literature search, code writing, editing LaTeX syntax for tables and figures, and language translation and polishing. The research ideas were developed by the authors without relying on AI for ideation. All experimental results were manually verified by the authors. The authors remain responsible for verifying citations, proofs, implementation details, and claims.

Ethics Statement

The work studies simulated tool-use agents and does not involve human subjects or private user data. The main safety concern is objective misalignment: process feedback can encourage shortcuts that appear successful while failing the user's task. We therefore use terminal task completion as the outcome metric and distinguish verifier predictions from ground-truth progress.

Reproducibility Statement

Section 5 specifies the benchmark splits, model families, baselines, and evaluation protocol. Theoretical assumptions and proofs appear in Section 3 and Appendix B. Appendix C documents data construction, training configurations, routing rules, and evaluation details for each stage of HARP. Code, training configurations, and evaluation scripts will be released upon acceptance.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated

mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Mohamed Aghzal, Gregory J. Stein, and Ziyu Yao. Why do LLM-based web agents fail? a hierarchical planning perspective. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32157–32180, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1483. URL https://aclanthology.org/2026.acl-long.1483/.

David Blackwell. Equivalent comparisons of experiments. *The Annals of Mathematical Statistics*, 24(2):265–272, 1953.

Justin Chen, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. MAGDi: Structured distillation of multi-agent interaction graphs improves reasoning in smaller language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7220–7235. PMLR, 2024. URL https://proceedings.mlr.press/v235/chen24ah.html.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *CoRR*, abs/2502.01600, 2025. doi: 10.48550/arXiv.2502.01600. URL https://arxiv.org/abs/2502.01600.

Zhengyu Chen, Teng Xiao, Huaisheng Zhu, Yige Yuan, Luan Zhang, and Jingang Wang. Co-harness: Co-evolving harnesses and model weights for LLM agents. *CoRR*, abs/2607.22688, 2026. doi: 10.48550/arXiv.2607.22688. URL https://arxiv.org/abs/2607.22688.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747/.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, 2006.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

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 of *Proceedings of Machine Learning Research*, pp. 15419–15462. PMLR, 2025. URL https://proceedings.mlr.press/v267/erdogan25a.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In *Advances in Neural Information Processing Systems*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Maryam Hashemzadeh, Elias Stengel-Eskin, Sarath Chandar, and Marc-Alexandre Côté. Subgoal distillation: A method to improve small language agents. In *Proceedings of the 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 1053–1075. PMLR, 2025. URL https://proceedings.mlr.press/v274/hashemzadeh25a.html.

Pengcheng Li, Zhengyang Zhang, Dongxu Zhang, Sui Huang, and Shaohua Ma. VICT: Verifier-instrumented credit tracing for long-horizon LLM agent reinforcement learning. *CoRR*, abs/2608.28128, 2026. doi: 10.48550/arXiv.2608.28128. URL https://arxiv.org/abs/2608.28128.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. MOPD: Multi-teacher on-policy distillation for capability integration in LLM post-training. *CoRR*, abs/2606.30406, 2026. doi: 10.48550/arXiv.2606.30406. URL https://arxiv.org/abs/2606.30406.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo García, and Mingyi Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *CoRR*, abs/2602.16165, 2026. doi: 10.48550/arXiv.2602.16165. URL https://arxiv.org/abs/2602.16165.

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

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, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. AdaPlanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2537. URL https://papers.nips.cc/paper_files/paper/2023/hash/b5c8c1c117618267944b2617add0a766-Abstract-Conference.html.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents. *CoRR*, abs/2603.08754, 2026. doi: 10.48550/arXiv.2603.08754. URL https://arxiv.org/abs/2603.08754.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jiDsk12qcz.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *CoRR*, abs/2605.06078, 2026. doi: 10.48550/arXiv.2605.06078. URL https://arxiv.org/abs/2605.06078.

Xiaomi LLM-Core Team. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026. URL https://arxiv.org/abs/2601.02780.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: On-policy skill distillation for agentic reinforcement learning. *CoRR*, abs/2606.26790, 2026. doi: 10.48550/arXiv.2606.26790. URL https://arxiv.org/abs/2606.26790.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://dblp.org/rec/conf/iclr/YaoZYDSN023.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. AgentEvolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025. URL https://arxiv.org/abs/2511.10395.

Haiyue Zhang. Credit without ground truth: Auditing step-level credit assignment in LLM agents against executed replay. *CoRR*, abs/2608.19760, 2026. doi: 10.48550/arXiv.2608.19760. URL https://arxiv.org/abs/2608.19760.

Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents. *arXiv preprint arXiv:2605.07725*, 2026. URL https://arxiv.org/abs/2605.07725.

A APPWORLD FAILURE CASES MOTIVATING ROLE SEPARATION

Table 3 summarizes three cases drawn from AppWorld trajectory audits as qualitative evidence for the proposed failure decomposition. Each case isolates a different intervention: revise the plan, repair the execution, or change the recovery hypothesis. The first two examples occur as local faults inside trajectories whose final outcome is successful. Such local faults still matter: under a different executor or downstream state, the same error could cause terminal failure.

Table 3: Three AppWorld trajectory patterns that motivate separating Planner, Executor, and Repairer supervision. The snippets are shortened for readability while preserving the causal distinction.

Role	Task and observed behavior	What the trace shows	Required correction
Planner	**Task:** Follow artists of reggae-genre songs in my playlists. **Observed:** The plan checks whether each *artist* has genre "reggae," then follows those artists.	The executor follows the plan faithfully: API calls succeed and five artists are followed. The plan nevertheless selects the wrong set because the task predicate is on the song, not on the artist. Independent audit: planner fault; executor fault absent.	Rewrite the predicate: playlist song → song genre = reggae → song artist IDs.
Executor	**Task:** Like Venmo transactions from yesterday or today involving coworkers. **Observed:** The plan correctly computes the dates and filters sender/receiver identities.	The generated code imports `datetime` but uses `timedelta`, producing `NameError`. Keeping the plan fixed and adding the missing import yields 15 transaction IDs, so the error is localized to execution.	Repair the tool code or arguments while preserving the valid plan.
Repairer	**Task:** Add Spotify-recommended R&B songs released this year to a new playlist. **Observed:** After an empty result, it searches titles and artist names for the literal string "R&B."	The same code is emitted four times; every observation is `[]`, every reward is 0, and the branch never terminates. The teacher-generated first repair and the later planner-generated repairs share the same action, so failure does not trigger a new diagnosis.	Update the failure hypothesis and inspect the song's structured genre/release fields before retrying.

Why these cases matter. The three traces separate errors that a scalar terminal score can conflate: executing an invalid plan can look locally successful, a valid plan can fail for a one-line implementation defect, and repeated retries can make no progress even after the original error is observable. They therefore provide a compact motivating example for HARP's role-conditioned routing and for the verifier signals used to decide when a Repairer intervention is warranted.

B PROOFS AND ADDITIONAL DETAILS

Scope of the analysis. The results in this appendix are Bayes-risk and probabilistic statements under explicit assumptions about information channels, stationary renewal behavior, and imitation error. They characterize when structured feedback and on-policy distillation can help. Whether the implemented HARP pipeline satisfies these conditions, and at what total training cost, is examined empirically in Section 5 and discussed in Section 6.

B.1 PROOF OF PROPOSITION 1

Fix the common context C, the finite intervention alphabet, and the joint distribution of (U, C, Z). For any probabilistic predictor ν and channel F,

$$\mathbb{E}[-\log \nu(U \mid C, F)] = H(U \mid C, F) + \mathbb{E}_{C,F}[\mathrm{KL}(P(U \mid C, F) \,\|\, \nu(\cdot \mid C, F))]\,. \tag{8}$$

The KL term is nonnegative and vanishes for the true posterior, proving $\mathcal{L}_F^\star = H(U \mid C, F)$. Because $R = g(Z, C)$, conditioning on Z, C already determines R. Therefore

$$\mathcal{L}_R^\star - \mathcal{L}_Z^\star = H(U \mid C, R) - H(U \mid C, Z, R) = I(U; Z \mid C, R).$$

This proves Eq. 1; the gap is strictly positive exactly under the non-sufficiency assumption. For Eq. 2, every aggregated-channel rule δ_R can be implemented with Z by setting $\delta_Z(C, Z) = \delta_R(C, g(Z, C))$. Taking the infimum over all structured-channel rules establishes weak dominance. Additional information can change the posterior without changing the optimal action, so a strict log-loss gap does not imply a strict gap for every decision loss.

Relation to the implemented supervision. The aggregated channel $R = g(Z, C)$ is a hypothetical matched comparison. In HARP, the sequential role-RL stages use scalar objectives, with an additional progress term in the Executor stage, while role identity enters through the selected policy, its training interface, and the routed teacher target. Applying Proposition 1 therefore requires specifying concrete channels that satisfy the aggregation relation. If Z is a deterministic function of the full visible context, or is conditionally independent of U given that context, then $\mathcal{G} = 0$; any benefit of structured feedback in that case would come from making the information easier for a finite model to use.

B.2 OPD SUCCESS UNDER FINITE INFERENCE BUDGETS

Renewal assumptions and the unbounded limit. The $N \geq 1$ stages are analysis units of comparable task progress; the Planner may produce local plans incrementally. A fresh stage begins with a local plan; $p \in [0, 1]$ is its feasibility probability and $e \in (0, 1]$ is execution success conditional on feasibility. After either an infeasible initial attempt or an execution failure, a recovery episode produces a feasible retry with probability $r \in [0, 1]$. This is an effective probability for the complete recovery behavior, which may include multiple repair activations and their routing or verification errors. Failure of the episode means that this abstract stage never completes.

We assume that recovery returns to the same progress stage with the same conditional execution law, that all required stages have the same parameters, and that their renewal transitions are independent. A repaired plan may differ from the failed plan. These assumptions exclude nonstationary or correlated failures. First-step conditioning gives $q = e + (1 - e)rq$, followed by $s = pe + (1 - pe)rq$, proving Eq. 3. For $d = 1 - (1 - e)r > 0$, $J_\infty = s^N$. In the boundary case $e = 0$, success is zero, including at $(e, r) = (0, 1)$ where the closed form is undefined. When stage counts or difficulties vary, success is obtained by averaging the conditional success probabilities over tasks, which generally differs from s^N evaluated at average parameters.

A shared budget including recovery costs. One execution attempt costs one unit. Let ρ_c be the probability that a recovery episode succeeds and consumes $c \geq 0$ additional units, excluding the failed attempt and the next feasible execution. Define

$$K(z) = \sum_{c \geq 0} \rho_c z^c, \qquad K(1) = \sum_{c \geq 0} \rho_c = r.$$

We assume the same recovery-cost kernel after each failure, independent of subsequent renewal transitions. With environment interactions as the unit, c includes any interactions inside recovery; planning-only outputs do not consume that particular budget. Counting tokens, latency, or all model decisions requires a corresponding cost model. Successful recovery is assumed to have finite cost almost surely.

Let $Q(z)$ and $F(z)$ encode the joint probabilities of successful stage completion and its total cost, starting respectively from a feasible attempt and from a fresh stage. They are defective probability generating functions: their mass may be below one because a stage may never complete. Conditioning on the first execution gives

$$Q(z) = ez + (1 - e)zK(z)Q(z) = \frac{ez}{1 - (1 - e)zK(z)}, \tag{9}$$

$$F(z) = pez + (1 - pe)zK(z)Q(z) = \frac{ez[p + (1 - p)zK(z)]}{1 - (1 - e)zK(z)}. \tag{10}$$

For a shared budget of T units across N independent stages,

$$J_T = \sum_{\ell=0}^{T} [z^\ell] F(z)^N, \tag{11}$$

where $[z^\ell]$ extracts a power-series coefficient. Thus $J_T = 0$ for $T < N$, $J_T \le J_{T+1} \le J_\infty$, and $J_T \to J_\infty$ as $T \to \infty$. Setting $z = 1$ recovers the main-text result.

An equivalent recursion makes the finite-budget calculation explicit. Let $A_n(t)$ be success with n stages and t units remaining from a fresh stage, and $B_n(t)$ success when the current stage already has a feasible attempt. Set $A_0(t) = 1$ for $t \ge 0$ and $A_n(0) = B_n(0) = 0$ for $n \ge 1$. For $n, t \ge 1$,

$$A_n(t) = pe\, A_{n-1}(t-1) + (1-pe) \sum_{c=0}^{t-1} \rho_c B_n(t-1-c), \tag{12}$$

$$B_n(t) = e\, A_{n-1}(t-1) + (1-e) \sum_{c=0}^{t-1} \rho_c B_n(t-1-c). \tag{13}$$

Then $J_T = A_N(T)$. Every recursive term uses a smaller remaining budget, including when recovery has zero additional cost.

Simpler finite-retry special cases. If recovery only supplies a revised plan between counted executions, $K(z) = r$. Write $a = pe$ and $b = (1-e)r$. The successful completion-cost probabilities for one stage are $f_1 = a$ and $f_\ell = (1-a)re\, b^{\ell-2}$ for $\ell \ge 2$. With a separate cap of $M \ge 1$ executions per stage,

$$s_M = a + (1-a)re \sum_{j=0}^{M-2} b^j = a + (1-a)re \frac{1-b^{M-1}}{1-b}, \qquad J_{\text{cap}} = s_M^N, \tag{14}$$

where the sum is empty for $M = 1$. A separate cap differs from a shared budget: unused attempts cannot be transferred to another stage. For a shared budget with $p = 1$, counting the f recoverable failures before the Nth successful execution yields the familiar special case

$$J_T = e^N \sum_{f=0}^{T-N} \binom{N+f-1}{f} [(1-e)r]^f, \qquad T \ge N. \tag{15}$$

The final attempt must be successful; the binomial coefficient places the failures among the preceding attempts. When recovery incurs additional counted interactions, the general recursion above applies instead.

Capability comparisons and training resources. The unbounded stage success satisfies

$$\frac{\partial s}{\partial p} = \frac{e(1-r)}{d}, \qquad \frac{\partial s}{\partial e} = \frac{[p+(1-p)r](1-r)}{d^2}, \qquad \frac{\partial s}{\partial r} = \frac{e(1-pe)}{d^2}.$$

All are nonnegative. Since $0 \le s \le 1$, reducing an integer N also weakly increases s^N. Strict parameter improvements can leave success unchanged; for example, $r = 1, e > 0$ gives $s = 1$. Because $k = 1/N$ is tied to a fixed stage granularity, merging stages can also change p, e, r. At finite budgets, recovery probability and recovery cost jointly determine performance. Relating these capability parameters to a training budget requires accounting for expert training, verifier and router construction, teacher queries, and distillation, which we leave to empirical study.

Proof of the finite-horizon OPD retention bound. This result does not require the renewal assumptions above. Fix the same initial task distribution and environment for teacher π_T and student π_S. Both act on the same observable history and action space for at most H decision steps, with terminal reward $Y \in [0,1]$; pad early termination with absorbing states. Teacher control state must be reconstructible from that history for this comparison. Let $d_{\pi_S,t}$ be the final student's history distribution and define

$$\epsilon_t = \mathbb{E}_{h \sim d_{\pi_S,t}} [\mathrm{TV}(\pi_T(\cdot \mid h), \pi_S(\cdot \mid h))], \qquad \epsilon = \frac{1}{H} \sum_{t=1}^{H} \epsilon_t,$$

with $\mathrm{TV}(P,Q) = \frac{1}{2}\sum_a |P(a) - Q(a)|$ for discrete actions. Let $Q_t^T(h,a) \in [0,1]$ be expected terminal success after action a followed by the teacher. Telescoping over policies that use the student for a prefix and the teacher thereafter gives

$$J_H(\pi_T) - J_H(\pi_S) = \sum_{t=1}^{H} \mathbb{E}_{h\sim d_{\pi_S,t}} \left[\mathbb{E}_{a\sim\pi_T(\cdot|h)} Q_t^T(h,a) - \mathbb{E}_{a\sim\pi_S(\cdot|h)} Q_t^T(h,a) \right]. \tag{16}$$

Each absolute difference is bounded by the corresponding total variation because $Q_t^T \in [0,1]$. Therefore

$$|J_H(\pi_T) - J_H(\pi_S)| \le \min\{1, H\epsilon\}. \tag{17}$$

For a mixed-training baseline evaluated with the same horizon, the sufficient retention condition is consequently

$$J_H(\pi_T) - J_H(\pi_{\mathrm{mix}}) > H\epsilon \quad \Longrightarrow \quad J_H(\pi_S) > J_H(\pi_{\mathrm{mix}}). \tag{18}$$

All three success probabilities use the same finite horizon H, and a decision step includes every modeled policy output, which may be finer than one environment interaction. On-policy supervision targets histories induced by the student (Agarwal et al., 2024), which is the distribution on which ϵ is measured. Relating ϵ to the routed sampled-token objective in Eq. 7, whose role assignment uses completed units and whose segment weights depend on sampled content, would require a separate error analysis.

C Implementation Details

This appendix records implementation settings in the order of the Methodology subsections, following SFT cold start, Executor RL, Planner RL, Repairer integration, and OPD.

C.1 Initialization and Training Sequence

The original Qwen3-1.7B, Qwen3-4B, and Qwen3-8B models initialize the three student sizes. At each student size, the expert teacher uses size-matched Planner, Executor, and Repairer models.

An SFT cold start produces π_{P}^0 and π_{E}^0; all three model sizes share the same cold-start configuration.

The subsequent sequence is fixed. Starting from π_{E}^0, Executor RL uses the frozen π_{P}^0 and produces π_{E}^*. Starting from π_{P}^0, Planner RL then uses the frozen π_{E}^* and produces π_{P}^*. We add the trained Repairer π_{R}^* and assemble $(\pi_{\mathsf{P}}^*, \pi_{\mathsf{E}}^*, \pi_{\mathsf{R}}^*)$ with the phase classifier and V_0. This complete teacher is frozen during OPD. The separate student starts from the original base model; the role experts' SFT initialization is a teacher-training stage.

C.2 Rollout Collection and Progress Verification

The observation-based verifier dataset contains 9,622 training examples: 7,119 `progress`, 2,395 `no_progress`, and 108 `regression`. Its heldout set contains 1,001 examples: 736, 256, and 9 in the same order. The transitions are drawn from failure points in Router2 rollouts that pair the Planner and Executor. Training and heldout data have disjoint task IDs and original-input hashes. Official Test-N and Test-C tasks are excluded from verifier-data construction. A separate 491-example development diagnostic set is enriched for error states.

Labels were constructed with model-assisted semantic annotation and review using GPT-5.6-Luna with Extra High reasoning, without independent human adjudication. Insufficient-evidence cases receive a null label and are excluded, as are the 1,635 records lacking a paired observation.

We fine-tune the full Qwen3-0.6B backbone with a sequence-classification head. The only serialized model input is `input_messages`: the verifier system instruction and full causal history through the paired current observation, including action/observation indices and the actual executed code. The annotation JSON and its label, reason, and evidence fields are excluded. Serialization uses the chat template without a generation prompt.

Training uses seed 42, learning rate 10^{-5}, a cosine schedule, warmup ratio 0.05, weight decay 0.01, bf16, FlashAttention2, and gradient checkpointing on eight A800 80GB GPUs. Per-device batch

size is one with eight accumulation steps, giving global batch size 64. The run requests three epochs and records 450 optimizer updates (2.9842 epochs). TF32 is disabled. Inputs are not truncated: the model limit is 40,960 tokens, with observed training and heldout maxima of 24,963 and 24,553. Batches use length grouping and dynamic padding to a multiple of eight.

The fixed final checkpoint is reloaded for one heldout evaluation; the heldout set is not used for checkpoint selection or tuning. Its accuracy is $980/1001 = 97.9021\%$, macro-F1 is 0.8169, and balanced accuracy is 0.7659. Only three of the nine regression examples, all from a single task, are detected. V_0 remains frozen during expert RL and progress-aware teacher routing.

C.3 Sequential Reinforcement Learning of Planner and Executor

We first train π_{E}^0 with π_{P}^0 frozen to obtain π_{E}^*, then train π_{P}^0 with π_{E}^* frozen to obtain π_{P}^*. Each stage updates only the active role's tokens, leaving its partner, router, phase classifier, and V_0 fixed. The Executor stage uses Eq. 6, with binary official terminal TGC, unit outcome coefficient, and progress coefficient two. Ordinary executed transitions with paired observations receive $P(\texttt{progress}) - P(\texttt{regression})$. An executed action lacking the required code format instead receives -1. Protocol-failure, context-exhaustion, or generation-budget exits add a synthetic -1 entry even without an observation; this entry also contributes to the mean denominator. Infrastructure failures, unknown verifier results, missing terminal scores, or an empty score list invalidate the whole GRPO group, which is then discarded. The subsequent Planner stage uses only binary terminal TGC.

Both stages use GRPO with full-parameter updates under FSDP, without LoRA. Each stage performs 20 updates; every update samples 4 task groups with 8 rollouts each, for 32 trajectories per update.

The GRPO (AgentEvolver) baseline is trained from the corresponding base model with 8 rollouts per group and 32 task groups per global step, for 60 updates.

C.4 Learning a Repairer from Failed Rollouts

Candidate collection selects 750 environment states: 489 strict-rule candidates, 111 exploratory boundary states, and 150 normal controls. Six branches are replayed from each state, for 4,500 branches in total. The branches include the original continuation (B0), a prompted continuation (B1), and four teacher candidates (T1–T4), with the Executor frozen. Verifier prefiltering retains candidates whose mean probability of `progress` is at least 0.80 and exceeds B0 by at least 0.10, or that succeed within four steps while B0 does not. This filter uses mean $P(\texttt{progress})$, rather than the progress-minus-regression reward used in role RL.

Same-state execution and Codex-assisted semantic review then check actual repair behavior, plan/execution boundaries, and user constraints. The strict-v3 dataset contains 17 repair plans across 13 tasks, mixed with 34 ordinary planning examples. Inputs end at the error observation, exclude future observations and the extra B1 prompting hint, and supervise only the first accepted repair plan.

At each model size, the Repairer starts from its matching SFT-trained Planner π_{P}^0 and uses the same reviewed 51-example mixture of repair and ordinary planning examples. LoRA SFT uses rank 32, scaling $\alpha = 64$, dropout 0.05, learning rate 10^{-4}, four epochs, and 28 optimizer updates. The fitted Repairer is added to the sequentially RL-trained Planner and Executor before OPD.

C.5 Progress-Aware Routing

The phase classifier uses `FacebookAI/xlm-roberta-base` (Conneau et al., 2020), revision `e73636d4f797dec63c3081bb6ed5c7b0bb3f2089`. Its labels are PLAN, TRANSITION, and EXECUTION. Inputs contain preceding units and the current complete sentence or code unit. The rule-assisted `strict_causal_v2` annotation pipeline divides 90 task IDs into train/validation/test at 80/10/10 with seed 42. There are 46,737 training units, including 3,308 ambiguous units down-weighted to 0.5; validation has 5,432 strict units and excludes 419 ambiguous units; test has 5,667 strict units and excludes 404 ambiguous units.

Training uses AdamW, learning rate 2×10^{-5}, a linear schedule with 6% warmup, and three epochs. Maximum input length is 512 with left truncation. Eight A800 80GB GPUs use per-device training/evaluation batches of 16/32, giving training global batch size 128. Validation selects the model from epoch three, step 1,098. The strict test accuracy is $5642/5667 = 99.5588\%$ and macro-F1 is 0.993820. Table 4 reports its three-class confusion counts. This test set is separate from the official AppWorld test splits, and its labels come from the rule-assisted annotation pipeline.

Table 4: XLM-R phase-classifier confusion matrix on the strict heldout partition. Rows are reference labels and columns are predictions; entries are counts.

Reference / Prediction	Plan	Transition	Execution
Plan	1,405	19	1
Transition	5	1,244	0
Execution	0	0	2,993

Repair decisions are made by V_0, which selects the Repairer for the next planning turn. Because transition handling also depends on the current role, phase accuracy measures one component of the full switching procedure.

Router2 preserves the accepted token prefix at handoffs. The role-RL routing configuration specifies `planner_each_observation`, `before_action_only` Executor re-planning, a maximum of eight switches per turn, Planner-EOS handoff, and a required Executor action.

Router3 adds the observation-based V_0 decision described in Section 4.5. After a Repairer-to-Executor handoff it disables further re-planning within that assistant turn. A subsequent observation determines the next plan owner anew. Verifier exceptions, overlength inputs, and invalid labels are logged and fall back to the Planner.

C.6 On-Policy Distillation into a Single Model

We use our adaptation of the SOD codebase (Zhong et al., 2026). The teacher comprises the sequentially trained π^*_{P} and π^*_{E} together with the trained π^*_{R}. All three expert policies, the XLM-R phase classifier, and the final V_0 are frozen. Each student starts from the corresponding original Qwen3 base model. Teacher assembly is completed before any on-policy student updates.

The student samples its own AppWorld trajectories. The teacher-scoring routine assigns roles to completed sentence/code units, then scores their sampled tokens under the corresponding teacher and observed prefix. Let B_{ik} denote a contiguous assistant-token segment in the response mask of trajectory i. With ℓ^S, ℓ^T defined in Section 4, the enabled segment-weighting branch computes

$$d_{ik} = \frac{1}{|B_{ik}|} \sum_{j \in B_{ik}} |\ell^S_{ij} - \ell^T_{ij}|, \tag{19}$$

$$w_{i1} = 1, \qquad w_{ik} = \min\left\{ \frac{d_{i1} + \varepsilon}{d_{ik} + \varepsilon}, 1 + \delta \right\} \quad (k > 1). \tag{20}$$

We use $\varepsilon = 10^{-6}$, $\delta = 0.2$, and coefficient $\lambda = 1$ in Eq. 7. This ratio equals the implementation's telescoping product of successive disagreement ratios. All weights and log-probability differences are computed from the collected rollout before the subsequent policy update. Tool observations and padding are excluded.

Each OPD step samples 4 training tasks with 2 rollouts per task, giving 8 student trajectories per step, at sampling temperature 0.7. The student learning rate is 10^{-6}. Training runs for 15 steps and saves a checkpoint every 3 steps; the reported student is the checkpoint with the best development-set TGC.

C.7 Evaluation Details

At each model size, the main HARP entry is the OPD checkpoint with the best development-set TGC, trained with the size-matched teacher. All main-table methods are evaluated under AgentEvolver's AppWorld protocol with greedy decoding and at most 30 environment interactions, covering all 168 Test-N and 417 Test-C tasks.

In the component table, base role interfaces use the unmodified base policy, which differs from the SFT-cold-started $\pi^0_{\mathsf{P}}, \pi^0_{\mathsf{E}}$ used to prepare the experts. All results are from single evaluation runs.

For Base Planner + trained Executor, success counts (Test-N, Test-C) are $(3, 0)$ at 1.7B, $(22, 15)$ at 4B, and $(35, 32)$ at 8B. For trained Planner + base Executor, the corresponding counts are $(1, 4)$, $(18, 16)$, and $(26, 18)$. The denominators are 168 and 417 tasks, respectively; Table 2 rounds each percentage to one decimal place.

Repairer routing diagnostic. Table 5 uses a separate Qwen3-8B evaluation whose Router3 run predates the final handoff rule in Appendix C.5; its trajectories are therefore distinct from those scored in Table 2. Router2 uses the two-role Planner–Executor system; Router3 adds the trained 8B Repairer and the frozen Qwen3-0.6B V_0 routing decision. The same frozen verifier scores both sets of trajectories. These are full task rollouts, separate from the short candidate branches used to construct the Repairer's training data. Both configurations use greedy decoding, seed zero, and at most 30 environment interactions. These are routed-system diagnostics, separate from the distilled HARP results.

For each configuration, let $\mathcal{T}_{\mathcal{D}}$ contain its successfully scored post-action observations in split $\mathcal{D}$. The reported `no_progress` fraction is

$$\widehat{\eta}_{\mathcal{D}} = \frac{1}{|\mathcal{T}_{\mathcal{D}}|} \sum_{t \in \mathcal{T}_{\mathcal{D}}} \mathbf{1}\left[\arg\max_{c \in \mathcal{C}} V_0(F_t)_c = \texttt{no_progress}\right]. \tag{21}$$

Table 5: Percentage of successfully scored post-action observations labeled `no_progress` by the same frozen V_0 in an 8B routing diagnostic. Changes are after minus before, computed from unrounded rates.

Split	Without Repairer	With Repairer	Change (pp)
Test-N	43.39	39.24	−4.15
Test-C	54.88	45.61	−9.28

The no-progress fraction decreases by 4.15 and 9.28 percentage points on Test-N and Test-C, respectively. V_0 itself is unchanged; the change is in its predictions on the resulting trajectories. The metric pools scored observations across tasks and excludes `regression`, unlike the general Repairer trigger.

The with-Repairer counts are $1021/2602$ (Test-N) and $3409/7475$ (Test-C), where each ratio is flagged observations over successfully scored observations. Without the Repairer, they are $1228/2830$ and $4911/8948$ for Test-N and Test-C. Table 5 reports percentages and differences calculated from these unrounded ratios.

Both test configurations cover the same 168 Test-N and 417 Test-C task IDs, without filtering by task success. Neither configuration receives a `regression` prediction. Equation 21 pools all scored observations across tasks. Unscored observations are excluded, so denominators differ with rollout length.

D Future Directions

AppWorld provides challenging interactions within a bounded digital environment with explicit task objectives and state-based evaluation (Trivedi et al., 2024). Extending this approach to a research project that might take humans months would require a broader hierarchy. An advisor-like role could exercise research taste and foresight by choosing promising questions and judging their long-term value. A senior-engineer-like role could coordinate implementation, dependencies, and resources across the project. Each subproject could itself require a long sequence of planning, execution, and recovery. The three roles in HARP should therefore be viewed as one useful decomposition for the present setting, with deeper and task-dependent hierarchies left to future work.

A further direction is to represent role requirements continuously. Instead of selecting among a fixed set of experts, a controller could regress quantities such as the required planning horizon or scope

of coordination, and use them to condition a policy or weight several experts. Such a representation would need meaningful supervision and evaluation: categorical role names do not by themselves define an ordered regression target. Lower levels could learn practical skills from frequent local feedback, while higher levels would need feedback that reflects delayed strategic value. In particular, useful exploration may initially appear unproductive, so optimizing a short-term progress proxy too strongly could discourage the behavior needed for discovery.

Training higher levels would also require going beyond freezing an Executor and optimizing a single Planner. Global plans must coordinate parallel lines of work under shared objectives, dependencies, and resource constraints. At the same time, decisions must adapt sequentially as new evidence becomes available. A promising direction is to maintain a persistent shared representation of project goals and dependencies, support concurrent work on independent subproblems, and revise plans causally as observations arrive. Evaluating such systems would require testing both global coherence and responsiveness to new evidence over extended horizons. Understanding how to train this interaction between parallel coordination and sequential adaptation, and how much of it can be retained through single-model distillation, is a central direction for future work.