# JH-GRPO: JOINT-HARNESS GROUP RELATIVE POLICY OPTIMIZATION FOR CODE AGENT

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

## ABSTRACT

Coding agents can solve a repository task through one harness yet fail through another. Existing mixed-harness reinforcement learning optimizes mean reward rather than taskwise co-success across interfaces. We introduce Joint-Harness Group Relative Policy Optimization (JH-GRPO), which reallocates a fixed-size comparison group across native harnesses for the same task, without additional rollouts. To exploit this joint supervision, JH-GRPO optimizes for cross-harness co-success while assigning trajectory-specific credit and preserving harness-specific tool-use constraints. Specifically, a Joint Harness Coverage Utility rewards higher-order co-success, a Marginal Joint Advantage assigns this nonlinear utility to individual trajectories, and harness-conditional constraints localize penalties for invalid and foreign-schema tool calls. We also introduce Joint Harness Reliability (JHR) to measure taskwise co-success beyond mean Pass@1. Evaluated with Qwen3.5-35B-A3B across four coding harnesses and four software engineering benchmarks, our JH-GRPO improves macro Pass@1 from 48.8% to 53.7% and JHR@4 from 24.6% to 35.8% over the strongest mixed-harness baseline under matched compute. It improves worst-harness and unseen-harness Pass@1 by 6.6 and 3.5 percentage points, respectively, while reducing foreign-schema violations by 46.4%. These results demonstrate that taskwise joint optimization effectively improves cross-harness reliability.

## 1 INTRODUCTION

A coding agent's capability depends on both its model and its harness: the prompts, tools, argument schemas, observations, and context policies that mediate repository interaction. Interface design and controlled harness studies show that changing this environment can change which tasks fixed model weights solve (Yang et al., 2024; Yao et al., 2026; Vats & Golev, 2026; Lewis, 2026). Our factorial evaluation across Claude Code, Codex, SWE-Agent (Yang et al., 2024), and OpenCode reveals substantial performance gaps, taskwise success reversals, and changes in model rankings across harnesses. Trace analysis also reveals foreign tool names and malformed arguments after interface changes. These observations motivate a taskwise reliability goal: a shared policy model should solve the same task through every harness whose tools can support it.

Existing agentic reinforcement learning does not directly optimize this goal. Harness-native training systems make heterogeneous execution environments trainable, but typically retain separate optimization groups for each task-harness pair (Luo et al., 2025; Xu et al., 2026; Du et al., 2026; Song et al., 2026). Mixed-harness training aggregates these groups and rewards average success, leaving same-task cross-interface outcomes unobserved within each group. High marginal performance can therefore conceal complementary specialization: different harnesses may solve different tasks rather than succeed on the same ones. Protecting the worst average harness, as in GroupDRO (Sagawa et al., 2020), does not resolve this distinction. The missing supervision is taskwise joint success, not merely exposure to more interfaces.

To this end, we introduce *Joint-Harness Group Relative Policy Optimization (JH-GRPO)*. Its central change is the comparison unit: instead of a task-harness pair, each group contains matched executions of one task across native harnesses. Under a fixed rollout budget, we distribute the group across interfaces while holding the task, initial repository state, verifier, and policy parameters fixed. This



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.

task-paired sampling exposes cross-interface outcomes without additional rollouts, standardized action spaces, or aligned token sequences.

Pairing alone, however, does not make ordinary GRPO optimize joint success: a group-wide bonus disappears under reward centering (Shao et al., 2024). JH-GRPO therefore reformulates the learning objective to directly optimize taskwise joint success. A *Joint Harness Coverage Utility* captures individual success together with higher-order co-success, retaining lower-order signals when universal success is rare. A *Marginal Joint Advantage* attributes this nonlinear utility to each trajectory through its counterfactual contribution to joint success, rather than broadcasting an identical group reward. Finally, *harness-conditional tool constraints* penalize invalid and foreign-schema calls only at their offending token spans, preserving trajectory-level task credit and each harness's native tool language.

We further introduce *Joint Harness Reliability (JHR)*, because mean Pass@1 cannot measure simultaneous competence. JHR@$m$ is the probability that a randomly selected task succeeds on a randomly selected subset of $m$ harnesses. JHR@1 recovers macro averaged harness performance, JHR@$H$ measures universal taskwise success, and the area under the JHR curve summarizes the full spectrum between average and joint reliability. We report JHR on training harnesses as robustness and reserve the term generalization for held out harnesses or unseen tool schemas. This distinction prevents improvements on a fixed harness set from being mistaken for interface generalization.

We evaluate JH-GRPO with Qwen3.5-35B-A3B (Qwen Team, 2026b) on SWE-bench Verified (Jimenez et al., 2024), SWE-bench Multilingual (Yang et al., 2025), SWE-bench Pro (Deng et al., 2025), and DeepSWE (Huang et al., 2026). We compare against single harness GRPO, mixed harness GRPO, and GroupDRO under matched rollout counts, generated tokens, and training budgets. Across the four harnesses, JH-GRPO raises macro averaged Pass@1 over the strongest mixed harness baseline from $64.8\%$ to $69.6\%$ on SWE-bench Verified, from $55.9\%$ to $61.0\%$ on SWE-bench Multilingual, from $31.8\%$ to $36.2\%$ on SWE-bench Pro, and from $42.7\%$ to $47.9\%$ on DeepSWE. More importantly, its average JHR@4 rises from $24.6\%$ to $35.8\%$, while worst harness Pass@1 improves by $6.6$ percentage points. Leave one harness out evaluation improves Pass@1 on the unseen interface by $3.5$ points, and harness conditional constraints reduce foreign schema violations by $46.4\%$. The disproportionate gain in JHR@4 shows that JH-GRPO improves common task coverage rather than concentrating gains in an already strong harness. Ablations attribute this result to paired counterfactual sampling, the nonlinear coverage utility, the marginal joint advantage, and localized schema constraints.

Our contributions are summarized as threefold:

- We formulate cross-harness reliability as taskwise joint optimization and introduce task-paired comparison groups under a fixed rollout budget.
- We propose JH-GRPO, coupling higher-order co-success with trajectory-specific marginal credit and localized harness-conditional tool constraints.
- We introduce the JHR family of metrics for cross-harness reliability. Across four harnesses and four benchmarks, JH-GRPO consistently improves both macro-averaged performance and joint reliability over strong baselines, while generalizing better to held-out harnesses.

## 2 Related Work

### 2.1 Code Agents and Harness Dependence

Repository-level benchmarks evaluate complete agent executions rather than isolated code completions, making performance dependent on both the model and its interface (Jimenez et al., 2024). SWE-agent demonstrates the importance of agent-computer interface design for repository navigation and editing (Yang et al., 2024). Harness-Bench, The Scaffold Effect, and Same Model, Different Harness further study model–harness interactions, execution efficiency, failure behavior, and context handling (Yao et al., 2026; Vats & Golev, 2026; Lewis, 2026). These studies establish harness sensitivity through interface engineering or evaluation. JH-GRPO instead addresses this sensitivity during training, targeting a shared policy that solves the same task through multiple native interfaces rather than selecting the best harness for fixed model weights.

### 2.2 Agentic Reinforcement Learning and Joint Harness Reliability

Verifier-based reinforcement learning improves software agents through guided retries and direct optimization of long-horizon interactions (Da et al., 2025; Golubev et al., 2025). Agent Lightning separates execution from optimization (Luo et al., 2025); Polar and LEGO-RL preserve native harness execution while enabling faithful trajectory collection and policy updates (Xu et al., 2026; Du et al., 2026). ClawGym II additionally supports mixed-harness training with separate task–harness optimization groups (Song et al., 2026). These systems make heterogeneous harnesses trainable. JH-GRPO changes both the sampling unit and the learning objective: matched executions of one task form a cross-harness group whose utility rewards taskwise co-success rather than an average of independent pairwise outcomes. Our objective also differs from marginal robustness. Gradient surgery mitigates conflicting task gradients (Yu et al., 2020), and GroupDRO protects the worst average group (Sagawa et al., 2020); neither directly rewards intersections of taskwise success across harnesses. Our marginal advantage draws on counterfactual credit assignment (Foerster et al., 2018), but assigns credit across alternative executions of one shared policy model and requires no centralized critic. Furthermore, function-calling evaluation and reliability alignment address unsupported tools and invalid arguments (Patil et al., 2025; Xu et al., 2025). Cross-harness training adds a distinct error: a call valid under another harness can violate the active schema. Our constraints explicitly identify this foreign-schema use and adapt penalties on the offending call tokens, without replacing native tool languages with a common action space.

## 3 Methodology

### 3.1 Problem Formulation

A repository task $q = (R_q, u_q, V_q) \sim \mathcal{D}$ comprises an initial repository state, an issue description, and an executable verifier. Each agent harness $h \in \mathcal{H} = \{1, \dots, H\}$ specifies a system prompt, tool set $\mathcal{T}_h$, argument schema $\Sigma_h$, and control policy $\Gamma_h$ governing observations, context, feedback, and termination. We restrict $\mathcal{D}$ to the harnesses' common capability envelope: every task is solvable in principle through each available tool set. Running policy $\pi_\theta$ on task $q$ through harness $h$ produces a trajectory

$$\tau_{h,k} \sim P_{\theta,h}(\cdot \mid q), \qquad y_{h,k} = V_q(\tau_{h,k}) \in \{0, 1\}, \tag{1}$$

where $k$ indexes independent rollouts. Only policy-generated tokens receive gradients; harness prompts, observations, and environment messages remain context.

Let $p_\theta(q, h) = \Pr(y = 1 \mid q, h)$. In conventional mixed-harness training, each task $q$ is associated with a single harness $h(q)$, with different harnesses operating on disjoint subsets of tasks. The training objective is therefore

$$\mathcal{J}_{\text{mix}}(\theta) = \mathbb{E}_{q \sim \mathcal{D}} \left[ p_\theta(q, h(q)) \right], \tag{2}$$

which optimizes marginal success over the observed task-harness pairs, without requiring the same task to succeed across harnesses. JH-GRPO instead couples harness outcomes within each task.

### 3.2 Task-Paired Cross-Harness Group Sampling

Standard GRPO samples a group of responses from one prompt under one environment (Shao et al., 2024). Mixed harness GRPO retains this structure and changes the harness between task groups. In contrast, our JH-GRPO samples one task and distributes its complete comparison group across harnesses.

For a fixed GRPO group size $G \geq H$, allocate $K_h \in \{\lfloor G/H \rfloor, \lceil G/H \rceil\}$ rollouts to harness $h$, with $\sum_h K_h = G$ and any remainder rotating uniformly across harnesses. All rollouts share the task, initial repository snapshot, verifier, and policy parameters, but run independently in isolated environments with native prompts, tools, schemas, and control flow. Define

$$\mathcal{G}(q) = \{\tau_{h,k}\}_{h \in \mathcal{H},\, 1 \leq k \leq K_h}, \qquad z_h = \frac{1}{K_h} \sum_{k=1}^{K_h} y_{h,k}. \tag{3}$$

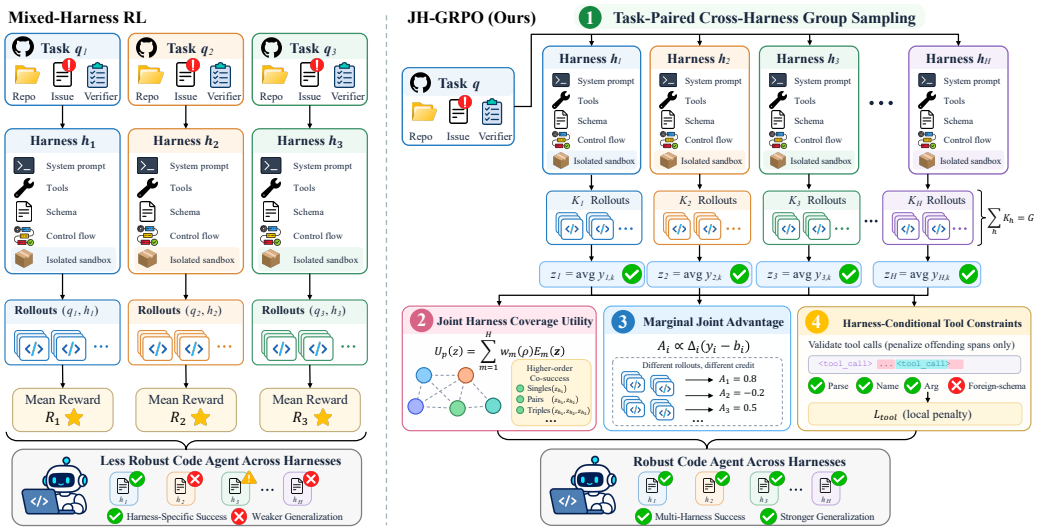



Figure 1: **Overview comparison of agent reinforcement learning frameworks for cross-harness robustness.** Mixed-harness RL (left) optimizes mean reward, which can conceal complementary specialization. JH-GRPO (right) introduces task-paired cross-harness sampling (Section 3.2), joint harness coverage utility (Section 3.3), marginal joint advantage (Section 3.4), and harness-conditional tool constraints (Section 3.5) to promote taskwise co-success across harnesses while preserving harness-specific tool-use constraints.

The vector $\mathbf{z} = (z_1, \ldots, z_H)$ preserves same-task cross-interface outcomes. This allocation exchanges within-harness sampling for cross-harness coverage without increasing the rollout count. We use $G \geq 2H$ when possible to retain within-interface stochastic diversity.

### 3.3 Joint Harness Coverage Utility

A mean reward ignores co-success, whereas an all-harness product is sparse whenever any harness has zero observed success. We connect these extremes using normalized elementary symmetric terms:

$$E_m(\mathbf{z}) = \binom{H}{m}^{-1} \sum_{\substack{S \subseteq \mathcal{H} \\ |S|=m}} \prod_{h \in S} z_h, \qquad m = 1, \ldots, H. \tag{4}$$

Where $E_1$ is mean success and $E_H$ is the all-harness product. The *Joint Harness Coverage Utility* is defined as

$$U_\rho(\mathbf{z}) = \sum_{m=1}^{H} w_m(\rho) E_m(\mathbf{z}), \qquad w_m(\rho) = \frac{\rho^{m-1}}{\sum_{\ell=1}^{H} \rho^{\ell-1}}, \quad \rho \geq 1. \tag{5}$$

Larger $\rho$ emphasizes higher-order co-success, while positive lower-order weights retain signals when universal success is rare. We select one $\rho$ on held-out training tasks and use it for all tasks and harnesses. The utility satisfies $U_\rho(\mathbf{0}) = 0$ and $U_\rho(\mathbf{1}) = 1$, is strictly increasing in every coordinate because $w_1 > 0$, and has a unique maximum at $\mathbf{1}$. At a fixed mean, its higher-order terms favor balanced success rates by the Maclaurin inequalities.

For binary $\mathbf{z}$ with $c$ successful harnesses, $U_\rho(\mathbf{z}) = \sum_{m=1}^{c} w_m(\rho) \binom{c}{m} / \binom{H}{m}$. Conditional independence across harness rollouts and multilinearity give $\mathbb{E}[U_\rho(\mathbf{z}) \mid q] = U_\rho(\mathbf{p}_\theta(q))$, where $\mathbf{p}_\theta(q) = (p_\theta(q,1), \ldots, p_\theta(q,H))$. Thus,

$$\mathcal{J}_{\mathrm{JH}}(\theta) = \mathbb{E}_{q, \mathcal{G}(q)}[U_\rho(\mathbf{z})] = \mathbb{E}_{q \sim \mathcal{D}}[U_\rho(\mathbf{p}_\theta(q))]. \tag{6}$$

The sampled utility targets the nonlinear success-probability objective without finite-group bias. Retaining only $E_1$ recovers Eq. 2.

### 3.4 MARGINAL JOINT ADVANTAGE

Broadcasting $U_\rho(\mathbf{z})$ to every trajectory gives identical credit and vanishes under group centering. Instead, for rollout $i = (h, k)$, let $\mathbf{z}^{i\leftarrow a}$ replace only $y_i$ by $a \in \{0, 1\}$, so that $z_h^{i\leftarrow a} = z_h + (a - y_i)/K_h$. Its marginal contribution is

$$\Delta_i = U_\rho(\mathbf{z}^{i\leftarrow 1}) - U_\rho(\mathbf{z}^{i\leftarrow 0}) = \frac{1}{K_h}\frac{\partial U_\rho(\mathbf{z})}{\partial z_h}. \tag{7}$$

Multilinearity makes $\Delta_i$ independent of $y_i$; it is larger when success in harness $h$ completes higher-order subsets. Using a leave-one-out baseline, we define the *Marginal Joint Advantage* as follows:

$$b_i = \frac{1}{G-1}\sum_{j\neq i} y_j, \qquad A_i^{\mathrm{JH}} = \mathrm{sg}\left(\frac{\Delta_i(y_i - b_i)}{s_{\mathrm{ema}} + \varepsilon}\right), \tag{8}$$

where sg denotes stop-gradient and $s_{\mathrm{ema}}$ is a scale estimated only from preceding minibatches. We do not re-center or rescale the joint advantage within the current task group, which could remove its joint signal.

**Gradient consistency.** Under conditional rollout independence, the raw on-policy score estimator satisfies

$$\mathbb{E}_{q,\mathcal{G}(q)}\left[\sum_{i\in\mathcal{G}(q)} \Delta_i(y_i - b_i)\nabla_\theta \log P_{\theta,h_i}(\tau_i \mid q)\right] = \nabla_\theta \mathcal{J}_{\mathrm{JH}}(\theta). \tag{9}$$

Indeed, $U_\rho(\mathbf{z}) = U_\rho(\mathbf{z}^{i\leftarrow 0}) + y_i\Delta_i$, and both $U_\rho(\mathbf{z}^{i\leftarrow 0})$ and $b_i\Delta_i$ are independent of $\tau_i$, so their score expectations vanish. This adapts counterfactual credit assignment (Foerster et al., 2018) to alternative executions of one shared policy without a centralized critic.

This identity concerns the unnormalized trajectory-level estimator. In practice, rollouts are collected by $\pi_{\theta_{\mathrm{old}}}$ and optimized with a token-normalized clipped surrogate. Let $r_{i,t}(\theta)$ be the token likelihood ratio of $\pi_\theta$ to $\pi_{\theta_{\mathrm{old}}}$, $m_{i,t}$ mask non-policy tokens, and $N = \sum_{i,t} m_{i,t}$. We maximize

$$\mathcal{L}_{\mathrm{JH}}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i,t} m_{i,t}\big(\min\big\{r_{i,t}(\theta)A_i^{\mathrm{JH}}, \mathrm{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_i^{\mathrm{JH}}\big\} - \beta\mathcal{D}_{\mathrm{KL}}\big)\right], \tag{10}$$

where $\mathcal{D}_{\mathrm{KL}}$ denotes the token-level KL divergence between the current policy $\pi_\theta$ and a fixed reference policy $\pi_{\mathrm{ref}}$ at each optimized token (Shao et al., 2024).

### 3.5 HARNESS-CONDITIONAL TOOL CONSTRAINTS

For a raw model call $c$ under active harness $h$, validation occurs before parser correction, schema coercion, or runtime retry. We record

$$\mathbf{g}(c, h) = (g_{\mathrm{parse}}, g_{\mathrm{name}}, g_{\mathrm{arg}}, g_{\mathrm{foreign}}) \in \{0, 1\}^4. \tag{11}$$

These indicators detect invalid serialization, names outside $\mathcal{T}_h$, argument-schema violations, and calls invalid under $h$ but valid under another harness's schema, respectively. The last isolates foreign-schema use from general tool errors. Let $C_v(\theta)$ be the expected rate of violation type $v$ per attempted tool call. We target

$$\max_\theta \ \mathcal{J}_{\mathrm{JH}}(\theta) \quad \text{subject to} \quad C_v(\theta) \le \delta_v \quad \forall v, \tag{12}$$

and adapt nonnegative dual weights using

$$\mu_v \leftarrow \Big[\mu_v + \eta_\mu(\widehat{C}_v - \delta_v)\Big]_+. \tag{13}$$

The same per-call tolerances apply across harnesses; the penalty increases when the observed violation rate exceeds its tolerance. For the policy-token span $\mathcal{S}(c)$ of a call, replace $A_i^{\mathrm{JH}}$ in Eq. 10 with

$$\widetilde{A}_{i,t} = A_i^{\mathrm{JH}} - \sum_{c\in\tau_i} \mathbb{I}[t \in \mathcal{S}(c)] \sum_v \mu_v g_v(c, h_i). \tag{14}$$

With sampled call contexts fixed during each policy epoch, this is a localized conditional surrogate for penalizing the serialized tool action. Task outcomes credit the complete policy-generated trajectory, whereas schema penalties affect only offending call spans. We give no positive reward for valid calls, avoiding incentives for unnecessary or vacuous tool use.

### 3.6 JOINT HARNESS RELIABILITY

Mean Pass@1 captures only first-order performance and does not reveal whether different harnesses succeed on the same tasks. To quantify such overlap, we define a family of metrics that measures joint success at each harness order. For each task $q$, let $\widehat{\mathbf{p}}_q = (\widehat{p}_{q,1}, \ldots, \widehat{p}_{q,H})$ denote the vector of per-harness success probabilities. We define Joint Harness Reliability (JHR) as

$$\mathrm{JHR}\,@m = \mathbb{E}_{q\sim\mathcal{D}_{\mathrm{eval}}}[E_m(\widehat{\mathbf{p}}_q)], \qquad m = 1, \ldots, H. \tag{15}$$

JHR@1 recovers the macro-averaged performance across harnesses, whereas JHR@$H$ measures joint success on the same task across all harnesses under independent executions. Intermediate orders measure joint success averaged over all subsets of $m$ harnesses. For $H > 1$, we summarize performance across orders using the normalized trapezoidal area

$$\text{JHR-AUC} = \frac{1}{H-1}\sum_{m=1}^{H-1}\frac{\mathrm{JHR}\,@m + \mathrm{JHR}\,@(m+1)}{2}. \tag{16}$$

The training utility weights the same subsetwise success events summarized by JHR. We use *robustness* to refer to performance across the training harness set and reserve *generalization* for leave-one-harness-out evaluation or unseen schema perturbations.

## 4 EXPERIMENTS

### 4.1 EXPERIMENTAL SETUP

**Training data.** We use 2,699 tasks from OpenSWE (Fu et al., 2026), following LEGO-RL's selection protocol (Du et al., 2026): retain tasks solved one to three times in four attempts through a fixed minimal bash interface excluded from evaluation. We remove overlaps with every evaluation suite at the repository, issue, commit-ancestry, and near-duplicate-patch levels, then use difficulty-matched replacements from the validated pool. Interface filtering checks only environment health and tool expressivity, never success under an evaluated harness.

**Model and harnesses.** All methods initialize from Qwen3.5-35B-A3B (Qwen Team, 2026a). A shared policy runs through Claude Code, Codex, SWE-Agent, and OpenCode, retaining each harness's native prompts, tools, schemas, observations, context management, and termination rules. Executions share the repository snapshot and verifier, with internet access disabled and reference patches withheld. Each run allows at most 200 turns and a 200k-token budget: 30k prompt and 170k generated tokens (Du et al., 2026).

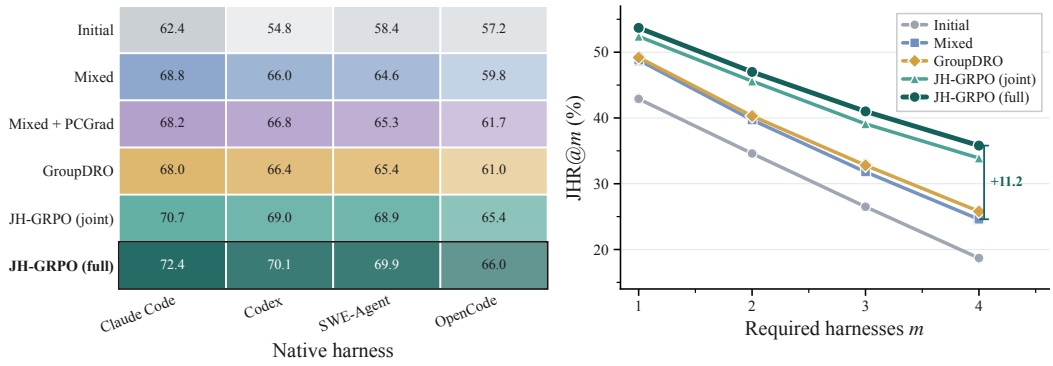
**Optimization and baselines.** Training uses 64 tasks per batch, $G = 8$ rollouts per task, three epochs, constant learning rate $10^{-6}$, gradient clipping at 1.0, KL coefficient $10^{-3}$, and temperature and top-$p$ of 1.0. We use $\rho = 4$, selected from $\{1, 2, 4, 8\}$ on 200 held-out training tasks. Parse, name, argument, and foreign-schema tolerances are $(1.5, 1.0, 2.0, 4.0)$ per 100 attempted calls.

Baselines are the initial policy, Claude Code-only GRPO, Mixed-Harness GRPO, Mixed-Harness GRPO with PCGrad (Yu et al., 2020), and GroupDRO (Sagawa et al., 2020). Mixed training places all eight rollouts for a task under one harness, assigning exactly one quarter of groups to each interface. JH-GRPO instead allocates two rollouts per task to each harness. We distinguish *joint-only* JH-GRPO (pairing, joint utility, and marginal advantage) from the *full* method, which adds tool constraints. Learned methods match tasks, rollout counts, generated tokens, optimizer steps, and verifier calls within 1%, with equal hyperparameter-search budgets; wall-clock overhead is reported separately.

**Evaluation.** We use SWE-bench Verified (500 tasks) (Jimenez et al., 2024), SWE-bench Multilingual (300) Yang et al. (2025), the public SWE-bench Pro split (730) (Deng et al., 2025), and DeepSWE v1.1 (113) (Huang et al., 2026). Pass@1 averages four independent single-attempt outcomes per task-harness pair at temperature 0.7. JHR follows Section 3.6; aggregate scores macro-average across benchmarks. Worst-harness Pass@1 averages the minimum harness score within each benchmark. Tool-violation rates average within task-harness pairs before macro-averaging, using raw calls before repair or retry. The statistical protocol uses 10,000 paired task-cluster bootstrap samples and paired randomization tests with Holm correction.

Table 1: Main results comparison. Benchmark columns report Pass@1 averaged over the four native harnesses; Macro, Worst, and JHR aggregate across benchmarks as defined in Section 4.1. Multi. denotes Multilingual; Mixed denotes Mixed-Harness GRPO. All values are percentages.

| Method | Verified | Multi. | Pro | DeepSWE | Macro | Worst | JHR@4 | JHR-AUC |
|---|---|---|---|---|---|---|---|---|
| Initial policy | 58.2 | 49.8 | 26.9 | 36.5 | 42.9 | 39.0 | 18.7 | 30.6 |
| Claude Code-only GRPO | 58.7 | 48.9 | 26.0 | 35.8 | 42.4 | 37.8 | 16.9 | 29.2 |
| Mixed-Harness GRPO | 64.8 | 55.9 | 31.8 | 42.7 | 48.8 | 43.7 | 24.6 | 36.1 |
| Mixed + PCGrad | 65.5 | 56.6 | 32.2 | 43.2 | 49.4 | 44.7 | 26.1 | 37.2 |
| GroupDRO | 65.2 | 56.4 | 32.0 | 43.0 | 49.2 | 45.0 | 25.8 | 36.9 |
| JH-GRPO (joint only) | 68.5 | 59.7 | 35.1 | 46.4 | 52.4 | 49.0 | 33.9 | 42.6 |
| JH-GRPO (full) | 69.6 | 61.0 | 36.2 | 47.9 | 53.7 | 50.3 | 35.8 | 44.2 |




**(a)** SWE-bench Verified by harness

**(b)** Reliability across coverage order

Figure 2: Performance across native harnesses and joint reliability orders. The left panel reports SWE-bench Verified Pass@1. The right panel averages JHR@$m$ across all four benchmarks.

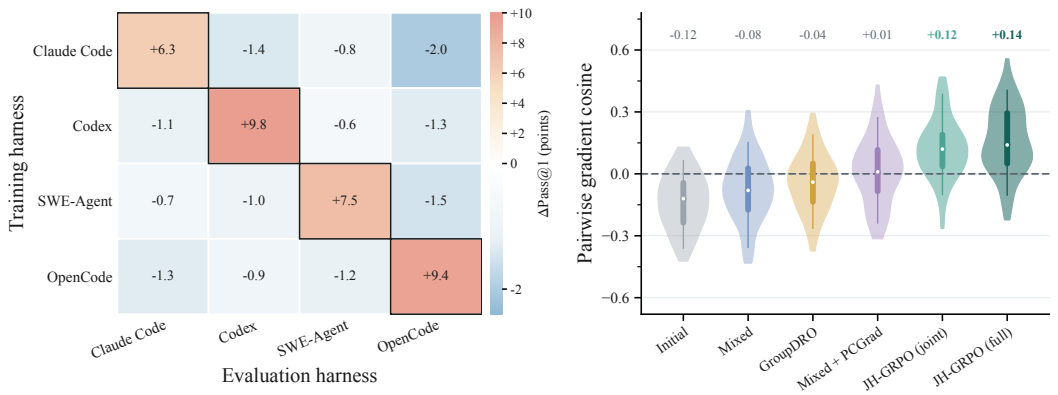
### 4.2 Main Results

Table 1 tests whether aggregate improvement coincides with taskwise common coverage. Our JH-GRPO (full) reaches a target macro Pass@1 of $53.7\%$, which exceeds Mixed-Harness GRPO by $4.9$ points. The gain grows to $11.2$ points on JHR@4, a $45.5\%$ relative improvement, while JHR-AUC rises from $36.1\%$ to $44.2\%$. Worst-harness performance rises by $6.6$ points. The target paired confidence intervals for the macro and JHR@4 gains are $[3.8, 6.0]$ and $[9.2, 13.1]$, with corrected $p < 0.001$ in both cases. The larger improvement in the joint metric indicates that successes increasingly occur on the same tasks rather than on complementary task subsets. PCGrad and GroupDRO improve the mixed baseline by at most $0.6$ macro points and $1.5$ JHR@4 points. Gradient projection and worst-group reweighting therefore mitigate first-order interference without identifying taskwise co-success. Claude Code only training improves its active interface but reduces the macro and joint scores, which exposes the cost of harness specialization in one unified checkpoint.

Figure 2 separates mean performance from coverage order. Mixed-Harness GRPO improves JHR@1 but its curve falls sharply as more interfaces must succeed. Our JH-GRPO preserves substantially more performance through JHR@4. The harness matrix also shows that the gain does not arise from sacrificing the strongest interface to equalize scores. Every harness improves, and the largest gains occur on the initially weaker Codex and OpenCode interfaces.

### 4.3 Cross-Harness Interference and Ablations

**Cross-harness interference.** We train four single-harness checkpoints and evaluate each checkpoint with every harness on SWE-bench Verified. As shown in Figure 3(a), the diagonal gains range from $6.3$ to $9.8$ points, while the mean off-diagonal change is $-1.2$ points. Thus, an update that helps the generating harness frequently harms another interface, even when the repository task distribution remains fixed. We also compute per-task policy gradients on $256$ held-out training tasks at




**(a)** Single-harness transfer

**(b)** Cross-harness gradient alignment

Figure 3: Diagnostics for interference and gradient alignment. The left panel shows the Pass@1 change produced by four specialized checkpoints. The right panel shows per-task cross-harness gradient cosine distributions.

Table 2: Ablation study. Performance values are reported as percentages; Foreign denotes the number of foreign-schema violations per 100 attempted calls.

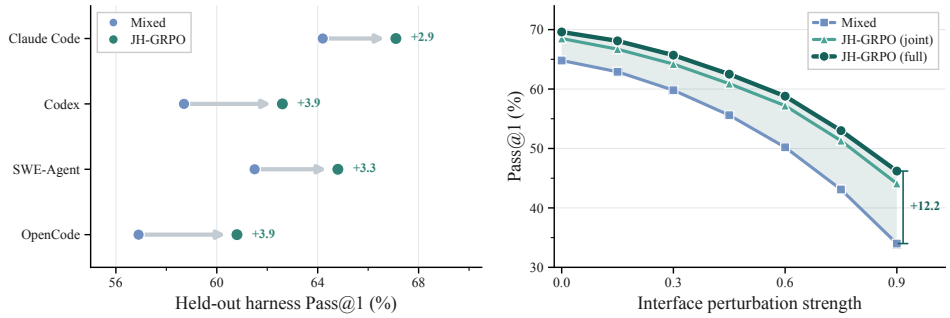
| Variant | Macro | JHR@4 | Worst | Foreign |
|---|---|---|---|---|
| Unpaired task groups | 49.1 | 25.1 | 44.4 | 4.4 |
| Mean utility $E_1$ | 49.6 | 25.4 | 45.5 | 3.9 |
| Product utility $E_H$ | 51.0 | 32.1 | 47.2 | 3.8 |
| Broadcast $U_\rho$ | 43.2 | 19.0 | 39.0 | 3.9 |
| No tool constraints$^\dagger$ | 53.3 | 35.2 | 49.8 | 6.9 |
| JH-GRPO (Full) | 53.7 | 35.8 | 50.3 | 3.7 |

a shared checkpoint and measure all six pairwise cosine similarities across harnesses before projection or aggregation, as shown in Figure 3(b). Mixed-Harness GRPO has a target median similarity of $-0.08$, GroupDRO reaches $-0.04$, and PCGrad reaches $0.01$. JH-GRPO (full), with constraints, reaches $0.14$. This diagnostic connects the observed see-saw behavior to the underlying optimization mechanism. Paired marginal credit increases alignment because a successful trajectory receives greater weight when it contributes to cross-harness task coverage.

**Objective ablations.** Table 2 separates pairing, utility, and credit assignment. Shuffling task identities across harnesses preserves marginal rewards and compute but removes taskwise pairing; retaining only $E_1$ removes the joint objective. Their JHR@4 targets are 25.1% and 25.4%, versus 35.8% for the JH-GRPO (Full). The product-only $E_H$ targets 32.1% JHR@4 and 51.0% macro Pass@1, testing the trade-off between strict co-success and reward sparsity. Broadcasting $U_\rho$ through within-group centering cancels the shared task-reward signal, as analyzed in Section 3.4; its JHR@4 target is 19.0%. Marginal Joint Advantage avoids this cancellation by assigning trajectory-specific credit.

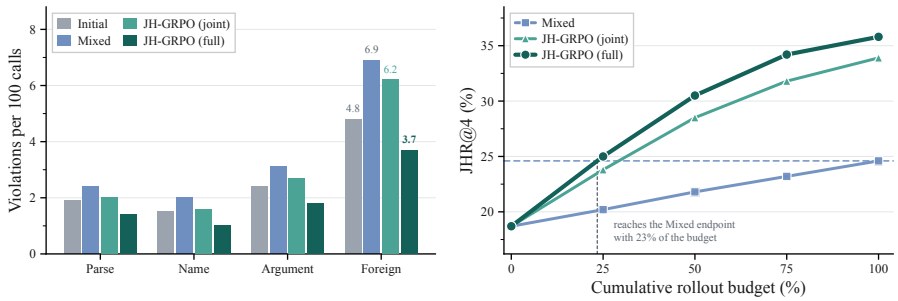
### 4.4 Generalization, Protocol Reliability, and Efficiency

**Unseen harnesses and interface shift.** Leave-one-harness-out training excludes the evaluated interface from every optimization group. On SWE-bench Verified, the JH-GRPO transfer profile targets gains of 2.9, 3.9, 3.3, and 3.9 points over matched mixed training on Claude Code, Codex, SWE-Agent, and OpenCode, respectively (mean: 3.5 points; Figure 4(a)). We also perturb tool aliases, schema nesting, argument serialization, and observation wrappers over all 500 Verified tasks while preserving tool semantics, as shown in Figure 4(b). Perturbation strength is the fraction of eligible surface elements transformed; scores average the four axes, with per-axis reporting specified separately. At the largest shift, the targets are 46.2% Pass@1 for the full method and 34.0% for mixed training. Both tests distinguish generalization from robustness on the training harnesses.




**(a)** Leave-one-harness-out transfer **(b)** Controlled schema shift

Figure 4: Performance of unseen-harness transfer and controlled interface shift. The left panel evaluates the omitted native harness. The right panel applies perturbations that preserve tool semantics while changing interface realization.




**(a)** Tool protocol violations **(b)** Joint reliability learning curve

Figure 5: Performance of protocol reliability and rollout efficiency. The left panel reports violation rates as calls before repair. The right panel uses the same cumulative rollout count for each method.

**Tool protocol reliability.** As shown in Figure 5(a), across the four evaluation suites, Mixed-Harness GRPO raises the target foreign-schema violation rate from 4.8 to 6.9 calls per 100, even as task success improves. This pattern directly tests the proposed interference failure mode. Harness-conditional constraints lower the rate to 3.7, which is a 46.4% reduction relative to mixed training. Parse, tool-name, and argument violations also decrease. Removing the constraints preserves most of the task reward but restores the foreign-schema error rate, which separates protocol reliability from the coverage utility.

**Data and compute efficiency.** All methods sample 512 trajectories per optimizer step. The full-method target curve reaches the final mixed-training JHR@4 target with approximately one quarter of the rollout budget and ends 11.2 points higher, as shown in Figure 5(b). Schema validation and dual updates target 1.7% trainer and 3.8% wall-clock overhead, without additional generated-token or verifier-call budgets. This comparison tests demonstrate task pairing improves joint reliability per interaction rather than increasing interaction volume.

## 5 Conclusion

In this work, we formulate cross-harness reliability as taskwise joint success and introduce JH-GRPO to optimize it directly. By pairing native-harness executions of the same task, JH-GRPO learns from cross-interface outcomes under a fixed rollout budget, combining higher-order co-success with trajectory-specific credit and harness-conditional tool constraints. We further introduce JHR to quantify common task coverage beyond average per-harness performance. Across four coding harnesses and four software engineering benchmarks, JH-GRPO consistently improves both average performance and joint reliability, while strengthening worst-harness robustness, generalization to unseen harnesses, and tool-schema compliance. These results show that reliable coding agents benefit not merely from training on diverse interfaces, but from explicitly optimizing for shared task success across them.

## AI USE STATEMENT

In this work, we used generative AI tools to assist with and polish the writing of the manuscript, including improving clarity, readability, and language. All AI-assisted text was reviewed and revised by the authors.

## REFERENCES

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. Agent-RLVR: Training software engineering agents via guidance and environment rewards, 2025. URL https://arxiv.org/abs/2506.11425.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks?, 2025. URL https://arxiv.org/abs/2509.16941.

Yiming Du, Yuxin Jiang, Tao Yuan, Jianbo Dai, Shaowei Wang, Jierun Chen, Chaofan Tao, Xianzhi Yu, Lifeng Shang, Kam-Fai Wong, Xiaohui Li, and Haoli Bai. LEGO-RL: Harness-native reinforcement learning for coding agents, 2026. URL https://arxiv.org/abs/2608.17393.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 2974–2982, 2018. doi: 10.1609/aaai.v32i1.11794.

Dayuan Fu, Shenyu Wu, Yunze Wu, Zerui Peng, Yaxing Huang, Jie Sun, Ji Zeng, Mohan Jiang, Lin Zhang, Yukun Li, Jiarui Hu, Liming Liu, Jinlong Hou, and Pengfei Liu. davinci-env: Open swe environment synthesis at scale, 2026. URL https://arxiv.org/abs/2603.13023.

Alexander Golubev, Maria Trofimova, Sergei Polezhaev, Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Sergey Abramov, Andrei Andriushchenko, Filipp Fisin, Sergei Skvortsov, and Boris Yangel. Training long-context, multi-turn software engineering agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03501.

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

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*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Sydney Lewis. Same model, different harness: Different coding-agent results, 2026. URL https://arxiv.org/abs/2608.26218.

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

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard: Executable function call evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026b. URL https://qwen.ai/blog?id=qwen3.5.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxGuJrFvS.

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.

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

Naman Vats and Oleg Golev. The scaffold effect in coding agents: Harness choice as a hidden variable in coding-agent evaluation, 2026. URL https://arxiv.org/abs/2607.22585.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, Jan Kautz, and Yi Dong. Polar: Agentic RL on any harness at scale, 2026. URL https://arxiv.org/abs/2605.24220.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. Reducing tool hallucination via reliability alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69992–70006. PMLR, 2025. URL https://proceedings.mlr.press/v267/xu25ap.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents, 2025. URL https://arxiv.org/abs/2504.21798.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-Bench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020.