How Well Do Models and Harnesses Work Together: Measuring Model–Harness Fit Beyond Task Scores

Anonymous authors
Paper under double-blind review

Abstract

LLM agents combine a model with an execution harness, so task scores reflect their joint behavior. These scores conflate model capability, harness-use capability, and task-solving support supplied by the harness. Because the same harness can impose different interaction demands on different models, we evaluate model–harness pairings through both task performance and operational validity. We introduce Valid Completion Rate (VCR) to measure normal termination with valid evaluator handoff, and model–harness Fit and Cross-Model Fit to summarize task performance achieved through valid completion. Using a common adapter that preserves native execution loops, we cross four general-purpose harnesses with multiple models across six benchmarks under matched conditions. Harness rankings vary across models and tasks, and task performance and valid completion can favor different pairings. Similar task scores can conceal distinct execution profiles, and low token use need not preserve outcomes. Cross-Model Fit identifies OpenCode as a stable aggregate leader within the evaluated panel, while model-specific preferences differ. These findings motivate joint assessment of task performance and valid completion alongside resource cost, training for transferable harness-use capability, and harness design adapted to the model and task.

1 Introduction

Classical views describe intelligent agents as goal-oriented systems that continuously perceive their environment, maintain internal state, select actions, and improve behavior through feedback (Wooldridge & Jennings, 1995; Wang et al., 2024). An LLM agent implements this paradigm by integrating a large language model (LLM) into an executable system for perception, reasoning, and action in external environments (Yao et al., 2023). At the system level, an LLM agent combines its model with an execution layer, the *agent harness*, that connects the model to the environment (Guo et al., 2026; Kwa et al., 2025; Yao et al., 2026). The harness provides mechanisms for observation processing, context management, tool and action interfaces, workflow control, state management, execution, evaluation, and error handling. By controlling how information is presented to the model and how its outputs are executed, the harness shapes agent behavior and task performance (Zhang et al., 2025b).

Under this system view, end-to-end benchmark performance reflects the joint behavior of the model–harness system rather than model capability alone (Yang et al., 2024; Yao et al., 2026). Yet most benchmarks remain task-centric, evaluating fixed model–harness configurations by task success or aggregate reward while treating the harness as part of the setup. These evaluations measure system success but offer limited insight into performance sources. Failures may reflect insufficient model capability, ineffective harness use, or harness limitations in observation processing, action interfaces, state management, or recovery (Ma et al., 2024; Zhu et al., 2026; Yao et al., 2026). An empirical issue study reports failures occurring only when particular model behaviors interact with harness parsing, control flow, or context handling (Chen et al., 2026). Conversely, high performance may partly reflect the harness's task-specific workflows or additional computation (Lou et al., 2026; Lee et al., 2026). Controlled studies show that changing harnesses can alter performance and rankings with models and tasks fixed (Zhu et al., 2026; Yao et al., 2026). Thus, a fixed configuration's task scores alone cannot distinguish the contributions of model capability, harness design, and their interaction.

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.

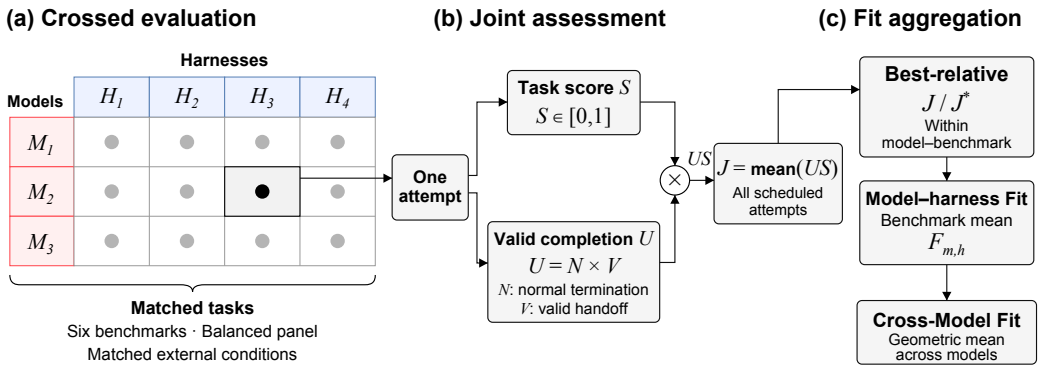

Figure 1: Overview of the model–harness evaluation framework.

The missing object of evaluation lies at the model–harness boundary. A model's *harness-use capability* depends on how effectively it can interact with a given harness: the same interface, action space, or feedback mechanism may be easy for one model to use but challenging for another, while the same model may behave differently under different harness designs (Yang et al., 2024; Yao et al., 2026). Accordingly, harness difficulty should be understood as a model-dependent interaction cost introduced by the harness's observation format, action interface, state management, control flow, and recovery mechanisms. When this cost exceeds the model's harness-use capability, the harness may prevent the model from realizing its underlying capabilities; when the harness provides extensive task-specific support, it may improve performance by contributing part of the task-solving process. Existing task-level evaluations conflate these effects into a single outcome, making it difficult to distinguish model capability from harness effects (Messick, 1995; Zhu et al., 2026; Yao et al., 2026). Therefore, evaluating agents requires examining both task achievement and valid completion, together with how these outcomes vary across model–harness pairings.

We develop metrics to analyze model–harness interaction through *task performance* and *operational validity*. Valid Completion Rate (VCR) measures how often attempts terminate normally with valid evaluator handoff, regardless of task correctness. Model–harness Fit and Cross-Model Fit summarize task performance from valid completions for individual models and across models. We analyze their diagnostic value and robustness, failure patterns, and resource use. We implement a common *agent adapter* for this study, preserving native execution loops under matched tasks and controlled execution conditions. We cross four general-purpose harnesses—DeepAgents, OpenCode, Pi CLI, and Codex CLI—with multiple models on six benchmarks: SWE-bench-45 (Jimenez et al., 2024; Chowdhury et al., 2024), Terminal-Bench 2 (Merrill et al., 2026), VitaBench (He et al., 2026), LOCA-Bench (Zeng et al., 2026), ClawMark (Meng et al., 2026), and AgentSkillOS (Li et al., 2026). Fixed native systems add complete-system references. Figure 1 summarizes the evaluation framework.

Harness rankings vary across models and benchmarks in both task performance and operational validity, which can favor different harnesses for one model. Their joint distribution reveals that similar success rates can conceal distinct execution profiles: clean runs may leave tasks unsolved, while non-clean runs may succeed or earn continuous-score credit. Pair-specific VCR deviations beyond model and harness averages also change sign across benchmarks. Resource use is likewise conditional: the lightest harness varies across models and tasks, and low token use need not preserve outcome. Low use can reflect premature termination; high use can include repeated failed actions. Cross-Model Fit identifies a stable aggregate leader on the evaluated panel, but model-specific preferences and intermediate ranks depend on its models and benchmarks.

These findings motivate joint assessment of task performance and operational validity: their joint distribution identifies task progress achieved through valid completion, while resource use describes its cost. Harness use should be trained and evaluated as a transferable capability across interfaces, including action construction, feedback use, recovery, and stopping. Harness design should adapt observation, recovery, and resource-control policies to the model and task.

2 Related Work

Harness design and optimization. LLM agents couple a model to an execution layer that constructs observations, exposes actions, maintains state, and closes the feedback loop. ReAct made this loop explicit, while SWE-agent and OpenHands showed how interfaces and runtime infrastructure shape agent behavior (Yao et al., 2023; Yang et al., 2024; Wang et al., 2025). A parallel line treats the surrounding system as an optimization target: GPTSwarm optimizes agent graphs, and ADAS and AFlow search over code-defined agent systems and workflows (Zhuge et al., 2024; Hu et al., 2025; Zhang et al., 2025a). More recent methods optimize constraints, context, full harness implementations, or editable agent code from execution feedback (Lou et al., 2026; Lee et al., 2026; Xu et al., 2026b; Zhang et al., 2026a;b; Lin et al., 2026). These methods show that the surrounding program can substantially change fixed-model performance. Their focus is optimization rather than separating lower interaction difficulty from added task-solving structure or pairing-specific compatibility.

Harness evaluation. Most agent benchmarks report task outcomes for a complete system (Liu et al., 2024; Mialon et al., 2024; Yao et al., 2025; Fu et al., 2026); AgentBoard adds progress and trajectory diagnostics but remains task-centered (Ma et al., 2024). Recent work addresses execution-layer confounding by crossing native harnesses and models or using shared adapters with standardized traces (Yao et al., 2026; Zheng et al., 2026; Wang et al., 2026a). Harness-Bench and harness-evolution analyses further show that harness effects are model-dependent (Zhang et al., 2026a). Other studies recommend reporting harness–model pairs, resource use, and failure profiles, and matching feedback budgets and held-out tasks (Vats & Golev, 2026; Wang et al., 2026b). Trace-level scaling work further distinguishes useful feedback from raw resource expenditure (Zhang et al., 2026c). These studies show that final success alone is insufficient, but they mainly rank or diagnose complete configurations. Our work measures task performance and operational validity across model–harness pairings, evaluates their joint contribution to effective operation, and quantifies pairing-specific deviations from additive component averages.

3 Preliminaries

Interpreting agent scores requires specifying the model, harness, and task conditions that produced them (Messick, 1995). We first describe how a model and harness form an executable agent, then distinguish task performance from operational validity and explain how their joint assessment characterizes effective operation. We also establish conditions for comparing model–harness pairings.

Model–harness configurations. An LLM agent combines a language model with an execution harness to interact with an external environment. The environment holds the task state, such as repository files or application records. The model produces text or tool-use outputs. The *agent harness* is the runtime that prepares model context and manages action execution (Guo et al., 2026; de Macedo, 2026). For a model M_m and harness H_h, indexed by m and h, the executable agent is $\mathsf{Agent}_{m,h} = H_h[M_m]$. At step t, the harness prepares model context c_t from the task and feedback x_t about environment state e_t. This context can include system and task instructions, tool descriptions, previous model outputs, and retained observations or summaries. For a coding task, observations might include a source-code excerpt, a command's output, or test results; context preparation selects and formats this information for the model (Yang et al., 2024). The model produces output $y_t \sim M_m(\cdot \mid c_t)$; the harness interprets requested tool calls and executes approved actions a_t, such as reading a file, applying an edit, or running a test. The resulting output or error is returned as feedback for subsequent steps, closing the loop (Yao et al., 2023; Yang et al., 2024). A model output may lead to zero or more actions and is distinct from an executed action. The environment and task success criteria remain outside the harness. Figure 2 summarizes this loop; Appendix H gives the formal interfaces.

Two dimensions of model–harness operation. Building on this execution loop, we distinguish two dimensions of model–harness operation: *task performance*, how well the final state satisfies the task's success criterion, and *operational validity*, whether execution terminates normally with an evaluator-consumable state, regardless of correctness. The same interface can impose different demands across models and tasks (Cronbach et al., 1972). Valid completion guarantees neither task success nor correct intermediate actions, and task credit may survive abnormal termination. We

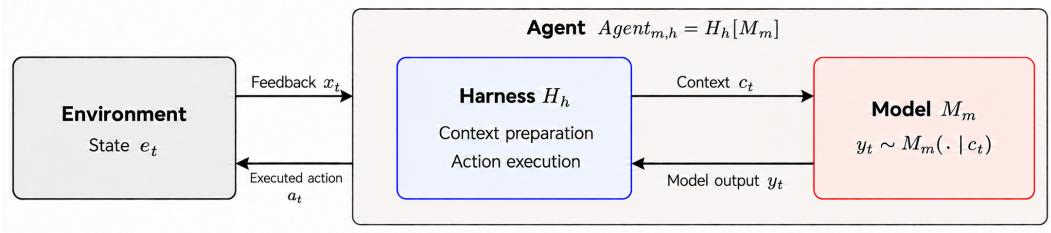

Figure 2: Model–harness composition and execution loop. The harness prepares model context and executes approved actions; model outputs y_t and executed actions a_t are distinct.

therefore assess the two dimensions separately for diagnosis and jointly to characterize *effective operation*: task performance achieved through valid completion of the same attempt.

Controlled comparisons. Crossed comparisons vary the model and harness independently, holding one fixed while changing the other. Task instances and external conditions, including tool access and resource limits, are matched to separate pairing effects from changes in task conditions (Fisher, 1935). Each harness retains its native execution loop, preserving the system being evaluated. Because harnesses bundle design choices, these comparisons do not isolate the causal effects of individual mechanisms (Holland, 1986).

4 Metric Design

We first measure task performance and operational validity, then compare harnesses on each dimension and jointly through Fit. Interaction residuals describe VCR deviations beyond model and harness averages, while resource metrics quantify execution cost. Cross-benchmark summaries use a *balanced panel*: the same models and harnesses evaluated in every combination on all included benchmarks. The panel and repetition count used in this study are specified in Section 5.

Task performance and operational validity. Official evaluator outcomes, averaged over R repetitions, measure task performance. For operational validity on task i and repetition r, let $N_{i,r}$ indicate normal termination and $V_{i,r}$ an evaluator-consumable submission or final state. A *clean run* achieves valid completion, $U_{i,r} = N_{i,r}V_{i,r} = 1$, irrespective of correctness. For each model–harness pair with benchmark task set $\mathcal{T}$, the *Valid Completion Rate* (VCR) is

$$\mathrm{VCR} = \frac{1}{|\mathcal{T}|\,R} \sum_{i \in \mathcal{T}} \sum_{r=1}^{R} U_{i,r}. \tag{1}$$

Mean VCR averages these rates equally across the included benchmarks for each model–harness pair. We report VCR and Mean VCR as percentages. Appendix C.3 separates abnormal termination from invalid handoff after normal termination.

Outcome Win Rate and VCR Win Rate. To compare each dimension across benchmarks, we score harness h against $h' \neq h$ within each model–benchmark block using the indicator function $\mathbf{1}(\cdot)$:

$$w_{h,h'} = \mathbf{1}(s_h > s_{h'}) + 0.5 \cdot \mathbf{1}(s_h = s_{h'}).$$

Here, s_h is the unrounded primary outcome for *Outcome Win Rate* or VCR for *VCR Win Rate*. Both metrics are obtained by averaging $w_{h,h'}$ equally over opponents and the included benchmarks, then converting to percentages. Across-model means use the balanced panel. Raw scores use benchmark-specific scales; win rates discard score margins.

Model–harness Fit and Cross-Model Fit. For a joint assessment, Fit compares task scores earned through valid completion across harnesses for the same model. Let $S_{i,r}^{m,h,b} \in [0,1]$ be the normalized official task score for model m, harness h, and benchmark b, retaining partial credit. Valid attempts

contribute their task scores; invalid attempts contribute zero. The mean over all scheduled attempts is

$$J_{m,h,b} = \frac{1}{|\mathcal{T}_b|R} \sum_{i \in \mathcal{T}_b} \sum_{r=1}^{R} U_{i,r}^{m,h,b} S_{i,r}^{m,h,b}. \tag{2}$$

For each model and benchmark, we divide J by the best observed harness score, $J^{\star}_{m,b} = \max_{h'} J_{m,h',b}$, following best-relative performance comparisons (Dolan & Moré, 2002). We average these ratios across benchmarks $\mathcal{B}$ to obtain model–harness Fit, reported as $100F_{m,h}$. Cross-Model Fit combines these model-specific scores with a geometric mean over the evaluated models $\mathcal{M}$:

$$F_{m,h} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{J_{m,h,b}}{J^{\star}_{m,b}}, \qquad \text{Cross-Model Fit}(h) = 100 \left(\prod_{m \in \mathcal{M}} F_{m,h} \right)^{1/|\mathcal{M}|}. \tag{3}$$

For example, a model–harness Fit of 80 means reaching 80% of the best observed score from valid completions for that model, averaged equally across benchmarks. Cross-Model Fit ranks harnesses for use across the evaluated models. The geometric mean penalizes weak Fit on individual models, favoring harnesses with consistently high Fit across models (Fan et al., 2023). Both scores range from 0 to 100 and are relative to the evaluated harnesses and tasks. If $J^{\star}_{m,b} = 0$, all ratios in that block are set to zero.

Interaction residuals. Let $v_{m,h}$ denote Mean VCR (%) on the balanced panel. We quantify pairing-specific deviations from the additive baseline as

$$\Delta_{m,h} = v_{m,h} - v_{m,\cdot} - v_{\cdot,h} + v_{\cdot,\cdot}. \tag{4}$$

Here, dots denote equal-weight averaging over the corresponding dimensions. The same centering within each benchmark gives residuals whose equal-weight mean equals this aggregate residual. Positive values indicate higher VCR than the additive baseline predicts. Residuals are panel-dependent, descriptive percentage-point deviations, not causal effects.

Resource metrics. Token Load is mean token consumption over all scheduled attempts, including failures. Tokens per Clean Run and Tokens per Success divide total consumption by the respective counts. Comparisons require 100% token coverage in matched model–benchmark blocks. Token-use wins favor lower load; outcome-preserving wins require no higher load and no lower official outcome, with at least one strict inequality. Pareto-frontier share measures the fraction of blocks where a harness is undominated; relative load is the geometric mean of within-block normalized loads. Tokens measure logical volume, not monetary cost (Appendix F.7).

Uncertainty and sensitivity. We use 2,000 paired task-cluster bootstrap replicates, stratified by benchmark and by environment within VitaBench. Sampled tasks retain all repetitions and model–harness observations. Each replicate recomputes outcome, operational validity, interaction, Fit, and matched resource statistics, including each Fit block's best-harness reference and Token Loads from provider-token records. We report pointwise percentile 95% intervals without multiplicity adjustment; cell-level resource summaries remain descriptive. Intervals reflect task variation, not decoding stochasticity or unseen-benchmark generalization. Leave-one-benchmark-out analyses assess dependence on benchmark composition.

5 Benchmarks and Experimental Setup

We apply the metrics in Section 4 to crossed evaluations of native harnesses and models on matched benchmark tasks. Appendix A gives task configurations and execution limits; Appendix G defines benchmark-specific scoring.

Evaluated harnesses, models, and benchmarks. We evaluate four general-purpose harnesses: DeepAgents (LangChain, n.d.), OpenCode (Anomaly, 2026), Pi CLI (Earendil Inc. and Contributors, n.d.), and Codex CLI (OpenAI, 2025). The core models are DeepSeek V4 Flash Preview (DeepSeek-AI, 2026b), MiniMax M3 (MiniMax, 2026), Kimi K2.7 (Moonshot AI, 2026), and Qwen3.6-35B-A3B (Qwen Team, 2026). Two fixed native references complement the crossed panels: GPT-5.5 xhigh

Table 1: Benchmark coverage and harness-sensitive interaction demands.

Benchmark	Tasks	Key interaction demands
SWE-bench-45 (Jimenez et al., 2024; Chowdhury et al., 2024)	45	Repository editing, testing, patch handoff
Terminal-Bench 2 (Merrill et al., 2026)	89	Stateful terminal interaction
VitaBench (He et al., 2026)	400	Multi-turn services, domain APIs
LOCA-Bench (Zeng et al., 2026)	75	Long-context planning, exact-state completion
ClawMark (Meng et al., 2026)	100	Multi-day workflows, state changes, multimodal inputs
AgentSkillOS (Li et al., 2026)	30	Skill selection, artifact handoff

(OpenAI, 2026) with Codex CLI, and DeepSeek V4 Pro 0813 (DeepSeek-AI, 2026c) with DeepSeek Harness (DeepSeek-AI, 2026a).

Benchmarks must require closed-loop action in an external environment or persistent artifact, support matched replay from prescribed states with an official evaluator, and exercise complementary harness-sensitive demands. Table 1 summarizes coverage. This designed suite supports conclusions conditional on the represented task demands.

Agent adapter. Our common agent adapter initializes tasks and connects models through each harness's native interface. It matches tool permissions, resource budgets, and termination conditions while preserving native prompts, observation and action interfaces, context management, and execution loops, including recovery and stopping. Runs use fresh task-scoped sandboxes without elevated privileges, isolated from evaluator internals, shared experiment storage, and other runs; protected credentials and resource controls remain outside. The adapter records final submissions or environment states, trajectories, resource use, and termination status in a common format. Original benchmark evaluators score results under unchanged task semantics and success criteria.

Crossed protocol and evaluation panels. Within each benchmark, all supported configurations run the same tasks with $R = 4$ repetitions; comparisons fix the task and repetition while varying one component. Four benchmarks use full 4×4 model–harness panels; ClawMark and AgentSkillOS use 3×4 panels because DeepSeek V4 Flash Preview lacks the required multimodal support. Cross-benchmark summaries use the common three-model, four-harness panel of 35,472 scheduled attempts. Fixed native systems provide matched complete-system references outside the crossed indices (Appendix B.3). Across all core panels, fixed references, and the AgentSkillOS stress condition, we analyze 52,168 trajectories.

6 Results and Implications

Three research questions guide our analysis and its implications for evaluation, model training, and harness design:

RQ1: Model–Harness Interaction. How do task performance, valid completion, and resource use vary across model–harness pairings and benchmarks?
RQ2: Cross-Model Fit. How do harnesses compare across models when task performance and valid completion are evaluated jointly?
RQ3: Metric Validity and Robustness. What information do the proposed metrics add beyond task scores, and how sensitive are their conclusions to aggregation choices and the evaluated models and benchmarks?

6.1 Findings on Model–Harness Interaction

Task performance and operational validity favor different pairings. For RQ1, harness changes can reverse model rankings within a benchmark. On SWE-bench-45, DeepSeek V4 Flash Preview outperforms MiniMax with Codex CLI, while MiniMax leads with OpenCode (Table 2). Harness preferences also depend on the metric: for Kimi, OpenCode leads Outcome Win Rate and Pi CLI leads VCR Win Rate; for Qwen, Codex CLI and OpenCode lead, respectively (Figure 3). Task credit and valid completion capture what a pair achieves and whether it completes the protocol validly, motivating joint assessment. Native systems provide matched competitiveness references: MiniMax–Codex exceeds both on LOCA-Bench, so neither provides a performance ceiling (Appendix B.3).

Table 2: Main outcomes (0–100; scoring in Appendix G). Best/Mean: Best-of-4/Mean-of-4; ClawMark: continuous scores; AgentSkillOS: Task Score (`skill_200`); VitaBench: equal-weight mean over four environments. Bold/underlined: best/second-best harness per core model and metric, including ties. Fixed references are excluded from highlighting and crossed analyses. Dashes: unevaluated; both DeepSeek models lack multimodal support for ClawMark and AgentSkillOS.

Model	Harness	SWE-bench-45		Terminal-Bench 2		VitaBench		LOCA-Bench		ClawMark		AgentSkill OS
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	
DeepSeek V4 Flash Preview	DeepAgents	40.00	31.11	**61.80**	**45.79**	69.00	38.00	40.00	20.67	–	–	–
	OpenCode	42.22	32.22	53.93	40.45	71.50	45.38	53.33	31.00	–	–	–
	Pi CLI	**46.67**	35.56	**61.80**	44.66	75.75	48.94	42.67	29.00	–	–	–
	Codex CLI	**46.67**	**36.11**	50.56	37.64	**77.00**	**49.06**	**76.00**	**50.67**	–	–	–
MiniMax M3	DeepAgents	55.56	31.11	46.07	21.63	77.25	**51.00**	33.33	24.00	75.06	54.78	80.07
	OpenCode	**82.22**	**62.78**	**55.06**	**39.61**	75.50	49.38	32.00	21.00	**82.39**	**69.62**	**93.49**
	Pi CLI	71.11	52.22	53.93	33.15	77.25	49.25	21.33	12.67	78.34	62.58	85.39
	Codex CLI	55.56	30.56	51.69	34.55	**77.50**	45.06	**93.33**	**87.00**	77.25	58.49	36.44
Kimi K2.7	DeepAgents	31.11	15.56	44.94	30.62	66.00	43.06	17.33	8.33	72.56	58.65	89.92
	OpenCode	**66.67**	**48.89**	53.93	39.04	68.50	**44.38**	42.67	27.67	**81.83**	**70.36**	84.94
	Pi CLI	51.11	42.22	**59.55**	**43.82**	68.75	40.25	50.67	34.33	12.08	11.93	**95.91**
	Codex CLI	28.89	10.56	48.31	31.74	**72.00**	41.44	**86.67**	**54.00**	12.20	11.96	55.66
Qwen3.6-35B-A3B	DeepAgents	31.11	15.00	40.45	26.40	59.25	27.69	14.67	8.67	68.58	52.71	74.01
	OpenCode	42.22	22.22	40.45	26.12	58.75	30.88	34.67	21.00	76.25	**63.62**	**76.12**
	Pi CLI	33.33	25.00	43.82	**28.65**	**59.50**	**31.19**	28.00	15.00	**76.91**	52.61	56.23
	Codex CLI	**44.44**	**26.67**	**47.19**	**28.65**	51.75	22.88	**70.67**	**36.33**	75.19	61.21	63.06
GPT-5.5 xhigh + Codex CLI		86.67	65.56	76.40	60.39	69.00	46.56	90.67	76.00	82.54	74.31	97.57
DeepSeek V4 Pro 0813 + DeepSeek Harness		88.89	70.56	62.92	46.91	79.25	54.13	90.67	73.33	–	–	–

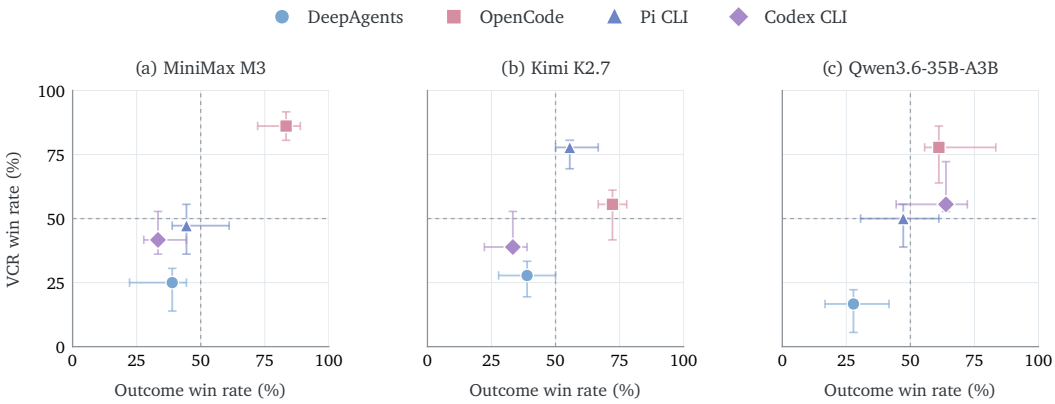

Figure 3: Task performance and operational validity on the balanced six-benchmark panel, summarized by Outcome Win Rate and VCR Win Rate. Each point represents a model–harness pairing; horizontal and vertical whiskers give the respective pointwise 95% task-cluster bootstrap intervals. Dashed lines: 50% (neutral).

Component averages hide pairing effects. MiniMax–OpenCode and Kimi–Pi exceed additive VCR baselines, with positive aggregate residuals whose pointwise bootstrap intervals exclude zero (Figure 6). Deviations vary by benchmark: Kimi–Codex is below baseline on SWE-bench-45 but above it on LOCA-Bench. Opposing deviations can cancel in aggregate, requiring benchmark-level profiles when interpreting model–harness interaction.

Joint outcomes link valid completion to task achievement. Joint outcomes connect task achievement and valid completion at the attempt level: non-clean runs may earn credit, while valid handoff need not solve a task (Figures 5 and 7). Kimi–Codex, for example, completes every ClawMark attempt cleanly yet performs poorly. Abnormal termination or invalid handoff motivates inspecting execution and finalization; low scores after valid completion motivate inspecting planning, action realization, and verification (Appendices C.3 and C.4).

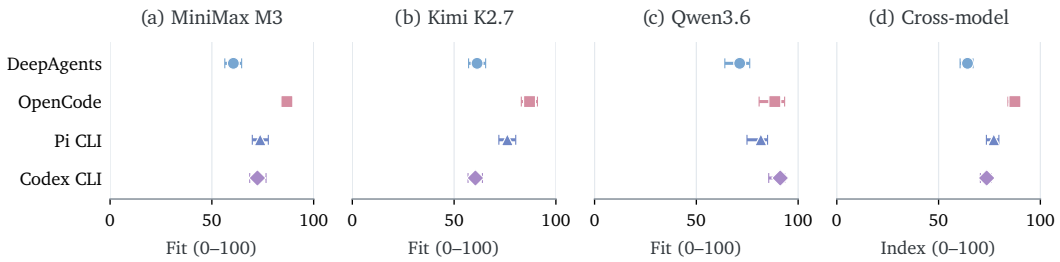

Figure 4: Model–harness Fit and Cross-Model Fit on the balanced six-benchmark panel (0–100). Whiskers: pointwise 95% paired task-cluster bootstrap intervals. Higher values retain more of the best observed joint score.

Cross-model fit complements model-specific choice. For RQ2, Figure 4 compares task scores from valid completions relative to each model–benchmark block's best observed harness. OpenCode has the highest Cross-Model Fit and similar Fit across all three models. Codex CLI leads for Qwen, but its lower Kimi Fit reduces its cross-model summary. The aggregate summarizes how consistently joint performance is retained across evaluated models; model-specific profiles identify preferences for a known deployment model. These support shared versus model-specific harness selection.

Resource costs qualify the choice of harness. Strong joint performance need not imply resource efficiency. Across complete primary comparison blocks, Pi CLI leads token-use win rate, while OpenCode leads outcome-preserving win rate and Pareto-frontier share (Table 28). On LOCA-Bench, Codex CLI has the highest outcome and lowest tokens per success for every model. On ClawMark, OpenCode outperforms DeepAgents using fewer tokens for all three models, while Kimi's low-load Pi and Codex runs score poorly. Low load can reflect early failure and high load can include repeated ineffective actions; cost must therefore be read alongside outcomes (Appendix F.7).

6.2 Metric Validation and Robustness

For RQ3, we examine diagnostic information, the contribution of each Fit component, and sensitivity to aggregation and evaluation coverage.

Diagnostic value beyond task scores. On SWE-bench-45, MiniMax–DeepAgents and MiniMax–Codex have nearly equal official success rates, with DeepAgents slightly ahead. Counting credit only from valid completions reverses their order: some DeepAgents successes are non-clean, whereas every Codex success is clean (Figure 5). The decomposition also retains partial credit on continuous benchmarks: non-clean attempts contribute nearly a third of Kimi–DeepAgents' ClawMark score (Appendix C.2). Yet the ClawMark audit finds gaps between intended and emitted actions in nearly clean configurations (Appendix C.4). Thus, VCR diagnoses protocol completion, while task scores measure what that interaction accomplished; both are needed.

Contribution of the Fit components. Table 3 compares the two matched win rates with three Fit variants. Replacing US with S removes the validity condition; arithmetic aggregation removes the geometric mean's preference for balanced model-specific scores. The validity condition most reduces DeepAgents' index, while geometric aggregation most affects Codex CLI, whose Fit varies most across models. All variants preserve the overall ordering. Fit therefore describes valid task-score retention and cross-model balance without changing the aggregate winner. Its components remain separately interpretable: normal completion of a zero-score attempt adds no joint credit, so VCR remains necessary to diagnose execution independently of success.

Stability under changing evaluation coverage. OpenCode remains first after removing any single model or benchmark. Intermediate ranks vary: removing Kimi favors Codex CLI over Pi CLI, while removing LOCA-Bench places Codex CLI below DeepAgents (Table 9). The full-panel paired bootstrap interval for the Pi–Codex index difference includes zero. Removing any single harness preserves the remaining ordering, though scores change with the best-harness references.

Table 3: Comparison metrics and Fit ablations on the common three-model panel. Outcome-only replaces US with S; arithmetic replaces the cross-model geometric mean. All benchmarks and models receive equal weight.

	Win rate (%)		Fit variants (0–100)		
Harness	Outcome	VCR	Outcome-only	Arithmetic	Cross-model
OpenCode	72.22	73.15	87.84	87.49	87.48
Pi CLI	49.07	58.33	78.20	77.20	77.13
Codex CLI	43.52	45.37	74.56	74.68	73.62
DeepAgents	35.19	23.15	68.15	64.38	64.20

These checks show a stable leader within the evaluated suite but panel-dependent intermediate ranks. Repeated-run and stress analyses further assess execution variability (Appendix C.5).

Selection across models. In leave-one-model-out selection, Fit and all comparison baselines choose OpenCode from the two remaining models. This matches the omitted model's highest Fit for MiniMax and Kimi, whereas Qwen favors Codex CLI (Table 10). Thus, a consistent shared choice can sacrifice a known model's preference. This retrospective diagnostic establishes neither a selection advantage for Fit over the baselines nor transfer beyond the small observed panel.

6.3 Implications for Evaluation, Training, and Harness Design

Evaluation. Ranking reversals motivate crossed evaluation on matched tasks. Use model-conditioned outcomes and VCR for diagnosis, and Cross-Model Fit for a harness shared across models. Retain benchmark-specific profiles and uncertainty to expose pairing effects and coverage-sensitive ranks. Joint outcomes, failure stages, and resource use separate successful task states from reliable, efficient execution. Native systems remain matched competitiveness references requiring the same process audits; their fixed components keep them outside crossed indices.

Model training. Native- and mixed-harness RL provide trajectories from diverse execution interfaces (Xu et al., 2026a; Song et al., 2026; Wei et al., 2026). Our ranking reversals motivate training for high model–harness Fit across interfaces, with strong task performance and valid completion under different harnesses. Training should pair the same tasks with multiple harnesses to develop transferable action realization, feedback use, recovery, and stopping. Our non-clean successes also motivate supplementing task rewards with feedback on execution validity. Under matched training budgets, evaluate outcomes, VCR, and Fit on held-out tasks and unseen harnesses to test generalization.

Harness design. Assess changes across models using Fit profiles and their outcome and VCR components before treating interfaces as broadly robust: aggregate gains can hide weaker pairings. Feedback should distinguish rejected from executed operations, and context compression should retain task-relevant state. Use model- and task-specific resource evidence to bound failed retries and reserve verification and handoff budgets. Controlled component ablations are needed to identify which mechanisms improve outcomes.

7 Conclusion

We presented a framework for evaluating model–harness interaction through task performance, valid completion, and resource use. Across six benchmarks, harness rankings depend on the model and task, and similar task scores can conceal different execution profiles. Model–harness Fit and Cross-Model Fit summarize task performance achieved through valid completion, while VCR and trajectory diagnostics expose execution failures that task scores alone miss. These findings support evaluating model–harness pairs and designing harnesses for reliable, efficient operation across models. They also motivate treating harness-use capability as a training objective, with the goal of sustaining strong task performance and valid completion across diverse execution interfaces, including harnesses unseen during training.

AI USE STATEMENT

We use generative AI tools assisted manuscript planning, drafting, language editing, and evidence-checking utilities. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Anomaly. OpenCode: Intro. Software documentation, 2026. URL https://opencode.ai/docs/. Updated September 24, 2026; accessed 2026-09-25.

Jingyi Chen, Songqiang Chen, Hengcheng Zhu, Jialun Cao, Jiasi Shen, and Shing-Chi Cheung. Understanding agent-reactive bugs at the model–harness boundary: An empirical study of LLM agent issue reports, 2026. URL https://arxiv.org/abs/2607.15684.

Neil Chowdhury, James Aung, Jun Shern Chan, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified. OpenAI, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Updated February 24, 2025.

Lee J. Cronbach, Goldine C. Gleser, Harinder Nanda, and Nageswari Rajaratnam. *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. Wiley, New York, 1972. ISBN 9780471188506.

Sanderson Oliveira de Macedo. What makes a harness a harness: Necessary and sufficient conditions for an agent harness, 2026. URL https://arxiv.org/abs/2606.10106.

DeepSeek-AI. DeepSeek Harness: Everything is a plugin. Open-source software, 2026a. URL https://github.com/deepseek-ai/deepseek-harness. Accessed 2026-09-25.

DeepSeek-AI. DeepSeek-V4-Flash model card. Hugging Face model card, 2026b. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash. Accessed 2026-08-17.

DeepSeek-AI. DeepSeek-V4-Pro GA Release. Official model announcement, 2026c. URL https://api-docs.deepseek.com/news/news260813/. Accessed 2026-08-18.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi: 10.1007/s101070100263. URL https://doi.org/10.1007/s101070100263.

Earendil Inc. and Contributors. Pi: Documentation. Software documentation, n.d. URL https://pi.dev/docs/latest. Accessed 2026-09-25.

Zimeng Fan, Nianli Peng, Muhang Tian, and Brandon Fain. Welfare and fairness in multi-objective reinforcement learning. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1991–1999, London, United Kingdom, 2023. International Foundation for Autonomous Agents and Multiagent Systems. URL https://www.ifaamas.org/Proceedings/aamas2023/pdfs/p1991.pdf.

Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

Lingyue Fu, Xin Ding, Linyue Pan, Yaoming Zhu, Shao Zhang, Lin Qiu, Xuezhi Cao, Xunliang Cai, Jiaxin Ding, Weiwen Liu, Weinan Zhang, and Yong Yu. CATArena: Evaluating evolutionary capabilities of code agents via iterative tournaments. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026. URL https://icml.cc/virtual/2026/poster/65908.

Jianyuan Guo, Zhiwei Hao, Chengcheng Wang, Cheng Fan, Tingzhang Luo, Hongguang Li, Ying Gao, Hefei Mei, Jiankun Peng, Rongjian Xu, Minjing Dong, Han Wu, Mengyu Zheng, Kai Han, Shiqi Wang, Chang Xu, and Yunhe Wang. From question answering to task completion: A survey on agent system and harness design, 2026. URL https://arxiv.org/abs/2606.20683.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Hui Su, and Xunliang Cai. VitaBench: Benchmarking LLM agents with versatile interactive tasks in real-world applications. In *International Conference on Learning Representations*, pp. 91654–91681, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/93fab021315170101c92e8330a56fbdb-Abstract-Conference.html.

Paul W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81 (396):945–960, 1986. doi: 10.1080/01621459.1986.10478354. URL https://doi.org/10.1080/01621459.1986.10478354.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, pp. 21344–21377, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

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 *The Twelfth International Conference on Learning Representations*, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://doi.org/10.1016/S0004-3702(98)00023-X.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long software tasks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 92213–92266. Curran Associates, Inc., 2025. doi: 10.52202/085713-3086. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/85069585133c4c168c865e65d72e9775-Abstract-Conference.html.

LangChain. Deep Agents overview. Software documentation, n.d. URL https://docs.langchain.com/oss/python/deepagents/overview. Accessed 2026-09-25.

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

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale, 2026. URL https://arxiv.org/abs/2603.02176.

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

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness, 2026. URL https://arxiv.org/abs/2603.03329. ICLR 2026 Workshop on Recursive Self-Improvement, poster.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362. Curran Associates, Inc., 2024. doi: 10.52202/079017-2365. URL https://doi.org/10.52202/079017-2365.

Fanqing Meng, Lingxiao Du, Zijian Wu, Guanzheng Chen, Xiangyan Liu, Jiaqi Liao, Chonghe Jiang, Zhenglin Wan, Jiawei Gu, Pengfei Zhou, Rui Huang, Ziqi Zhao, Shengyuan Ding, Ailing Yu, Bo Peng, Bowei Xia, Hao Sun, Haotian Liang, Ji Xie, Jiajun Chen, Jiajun Song, Liu Yang, Ming Xu, Qionglin Qiu, Runhao Fu, Shengfang Zhai, Shijian Wang, Tengfei Ma, Tianyi Wu, Weiyang Jin, Yan Wang, Yang Dai, Yao Lai, Youwei Shu, Yue Liu, Yunzhuo Hao, Yuwei Niu, Jinkai Huang, Jiayuan Zhuo, Zhennan Shen, Linyu Wu, Hannah Yao, Charles Chen, Cihang Xie, Yuyin Zhou, Jiaheng Zhang, Zeyu Zheng, Mengkang Hu, and Michael Qizhe Shieh. ClawMark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents, 2026. URL https://arxiv.org/abs/2604.23781.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong (Ryan) Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, pp. 40903–40986, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/444a3737adaee10d86ad2ef5f74468e6-Abstract-Conference.html.

Samuel Messick. Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientific inquiry into score meaning. *American Psychologist*, 50 (9):741–749, 1995. doi: 10.1037/0003-066X.50.9.741. URL https://doi.org/10.1037/0003-066X.50.9.741.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, pp. 9025–9049, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html.

MiniMax. MiniMax M3 model card. Hugging Face model card, 2026. URL https://huggingface.co/MiniMaxAI/MiniMax-M3. Accessed 2026-08-17.

Moonshot AI. Kimi-K2.7-Code model card. Hugging Face model card, 2026. URL https://huggingface.co/moonshotai/Kimi-K2.7-Code. Accessed 2026-08-17.

OpenAI. OpenAI Codex CLI. Open-source software, 2025. URL https://github.com/openai/codex. Accessed 2026-08-17.

OpenAI. GPT-5.5. Model documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.5. Accessed 2026-08-18.

Qwen Team. Qwen3.6-35B-A3B model card. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Accessed 2026-08-17.

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.

Haoning Wang, Mingxun Zhang, Chenyue Yu, Yingjun Shang, Xing Xie, Xia Hu, Guanchu Wang, and Na Zou. A^2E: An end-to-end agent auditing engine, 2026a. URL https://arxiv.org/abs/2608.07346.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, pp. 65882–65919, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a4b6ad6b48850c0c331d1259fc66a69c-Abstract-Conference.html.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents, 2026b. URL https://arxiv.org/abs/2607.12227.

Hongliang Wei, Xiaobing Tu, Yinggui Wang, Zhengxi Liu, Rongkun Xue, Jinkui Ren, Xiantao Zhang, Debin Zhao, and Xiaopeng Fan. HarnessBandit: Joint learnability-transferability scheduling for multi-harness agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2609.13739.

Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995. doi: 10.1017/S0269888900008122. URL https://doi.org/10.1017/S0269888900008122.

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, 2026a. URL https://arxiv.org/abs/2605.24220.

Tianshi Xu, Huifeng Wen, and Meng Li. Adapting the interface, not the model: Runtime harness adaptation for deterministic LLM agents, 2026b. URL https://arxiv.org/abs/2605.22166.

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, pp. 50528–50652, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, pp. 9965–10017, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

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.

Weihao Zeng, Yuzhen Huang, and Junxian He. LOCA-bench: Benchmarking language agents under controllable and extreme context growth. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026. URL https://icml.cc/virtual/2026/poster/64486.

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

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, pp. 104223–104294, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/aa5f5e6eb6f613ec412f1d948dfa21a5-Abstract-Conference.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, pp. 34040–34077, 2025a. URL https://openreview.net/forum?id=z5uVAKwmjf.

Shao Zhang, Xihuai Wang, Wenhao Zhang, Chaoran Li, Junru Song, Tingyu Li, Lin Qiu, Xuezhi Cao, Xunliang Cai, Wen Yao, Weinan Zhang, Xinbing Wang, and Ying Wen. Leveraging dual process theory in language agent framework for real-time simultaneous human-AI collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4081–4108, Vienna, Austria, July 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.206. URL https://aclanthology.org/2025.acl-long.206/.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. Scaling laws for agent harnesses via effective feedback compute, 2026c. URL https://arxiv.org/abs/2605.29682.

Mengyu Zheng, Kai Han, Boxun Li, Haiyang Xu, Yuchuan Tian, Wei He, Hang Zhou, Jianyuan Guo, Hailin Hu, Lin Ma, Chao Xu, Guohao Dai, Lixue Xia, Yunchao Wei, Yunhe Wang, and Yu Wang. Claw-SWE-Bench: A benchmark for evaluating OpenClaw-style agent harnesses on coding tasks, 2026. URL https://arxiv.org/abs/2606.12344.

Pengyu Zhu, Lijun Li, Yaxing Lyu, Qianxin Luo, Jingyi Yang, Yi Liu, Tingfeng Hui, Xinyu Yuan, Li Sun, Sen Su, and Jing Shao. UniACE: A unified framework for evaluating LLM agentic capabilities, 2026. URL https://arxiv.org/abs/2605.27898.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Benchmark Configuration Details

Four repetitions of each task are scheduled for every model–harness configuration unless specified otherwise. Each harness retains its native prompts and internal agent policy, with compatibility changes for unattended execution. The following paragraphs record the benchmark-specific task selection, environment, access policy, execution limits, and evaluator handoff used in our experiments.

Model serving and software versions. DeepSeek V4 Flash Preview was evaluated through its official API. MiniMax M3, Kimi K2.7, and DeepSeek V4 Pro 0813 were served locally with SGLang on a single node with eight NVIDIA H200 GPUs. The four crossed harnesses used DeepAgents Code 0.1.3, OpenCode v1.15.6, Pi CLI 0.79.3, and Codex CLI. The DeepSeek V4 Pro 0813 fixed reference used DeepSeek Harness SDK 0.1.0rc6. The GPT-5.5 xhigh–Codex CLI fixed reference was run through an official Codex subscription.

SWE-bench-45. SWE-bench Verified assigns each task a human-annotated difficulty label based on the time an experienced developer is expected to need to identify and implement a solution. We include every task whose estimate is at least one hour, yielding 45 of the 500 verified tasks: 42 labeled `1-4 hours` and three labeled `>4 hours`. The selected tasks span eight Python repositories, with 22 from Django, seven from SymPy, five from Sphinx, and 11 from the remaining five repositories. The subset includes all tasks with released difficulty estimates of at least one hour; "long" denotes estimated human completion time.

Each run starts from the prescribed base commit in the official containerized environment and receives the original GitHub issue description. We retain the task-specific regression tests and apply the official verifier to the final repository state. All harnesses operate on the same task images through the common adapter.

Terminal-Bench 2. We execute all 89 tasks through Harbor while retaining their original task definitions, containerized environments, verifier scripts, and per-task time limits. Every configuration starts from the same task environment and is assessed by the same task-specific verifier.

All 16 core configurations contain four distinct trajectories for each of the 89 tasks; both fixed system references use the same 89×4 support.

VitaBench. We use English task variants for 400 tasks across four environments (100 per environment), covering multi-turn services, domain-specific APIs, and cross-domain requests. Each task is repeated four times, giving 400 attempts per environment and model–harness configuration. A trajectory is limited to 300 interaction steps. We report the strict tool-permission setting: harness-native tools and external filesystem, shell, browsing, and delegation capabilities are disabled where applicable, so the agent can affect the simulated environment only through the VitaBench business APIs. DeepSeek-V3.2 serves as both the user simulator and trajectory evaluator. The simulator reveals task constraints through multi-turn interaction, and the evaluator applies the binary success criterion in Appendix G.

LOCA-Bench. We evaluate all 75 tasks in the 256K-context setting, exercising long-context state tracking and multi-tool plans. Each task specifies an initial application state and an exact target state to be reached through the available tools; the official evaluator compares the final state with that target.

ClawMark. We evaluate 100 tasks across 13 professional scenarios, covering multi-day workplace activity, silent state changes, and multimodal inputs. We apply a uniform 30-minute execution limit to every turn and every model–harness configuration. When a turn reaches this limit, we terminate the agent, evaluate the service state produced up to that point, and advance the task to the next in-universe working day. This preserves the benchmark's continuous workflow while applying the same per-turn budget to every configuration.

AgentSkillOS. We evaluate 30 tasks across five artifact categories, requiring skill selection, artifact production, and output-contract handoff. We preserve the original task prompts. The tested context budget accommodates the released 200-skill primary setting and a 900-skill stress setting containing all 19 task-required skills and 881 distractors drawn from the released skill pool. Each task has a 3,600-second wall-clock limit. Streaming harnesses additionally use a 900-second idle-timeout guard and the harness's retry policy where supported. Generated artifacts are scored with the official AgentSkillOS evaluator.

B Cross-Benchmark Comparison Tables

B.1 Outcome Win Rate

Table 4 reports the exact Outcome Win Rates summarized in Section 6.1.

Table 4: Six-benchmark Outcome Win Rates (%; Table 2). Columns: model-conditioned rates, head-to-head rates pooled across models, and the three-model mean. DA/OC: DeepAgents/OpenCode; ties receive half credit. Rates report tie-aware wins across the specified configuration comparisons.

Harness	By model			Head-to-head				Overall win rate
	MiniMax M3	Kimi K2.7	Qwen3.6 35B-A3B	vs. DA	vs. OC	vs. Pi	vs. Codex	
DeepAgents	38.89	38.89	27.78	–	22.22	33.33	50.00	35.19
OpenCode	**83.33**	**72.22**	61.11	**77.78**	–	**66.67**	**72.22**	**72.22**
Pi CLI	44.44	55.56	47.22	66.67	**33.33**	–	47.22	49.07
Codex CLI	33.33	33.33	**63.89**	50.00	27.78	52.78	–	43.52

B.2 Model-Conditioned Valid Completion

Table 5 reports the exact VCR Win Rate and Mean VCR values summarized in Section 6.1.

Table 5: Six-benchmark model-conditioned VCR Win Rate and Mean VCR (%). VCR Win Rate uses tie-aware comparisons; Mean VCR weights benchmarks equally. Bold/underlined: best/second-best harness within each column.

Harness	MiniMax M3		Kimi K2.7		Qwen3.6-35B-A3B	
	VCR win rate	Mean VCR	VCR win rate	Mean VCR	VCR win rate	Mean VCR
DeepAgents	25.00	60.60	27.78	59.69	16.67	79.31
OpenCode	**86.11**	**91.68**	55.56	83.09	**77.78**	**90.70**
Pi CLI	47.22	72.96	**77.78**	**87.94**	50.00	82.13
Codex CLI	41.67	71.88	38.89	78.26	55.56	89.48

B.3 Fixed System References

The fixed references provide complete-system baselines, while the crossed indices compare the factorial model–harness panel. The GPT-5.5–Codex reference covers all six benchmarks; DeepSeek V4 Pro 0813–DeepSeek Harness uses its standard preset on SWE-bench-45, Terminal-Bench 2, VitaBench, and LOCA-Bench. Both use the core panels' task sets, repetitions, external controls, and evaluators, including strict tool permissions on VitaBench.

GPT-5.5–Codex wins most matched outcome comparisons across the six benchmarks, but does not lead every benchmark in either outcome or VCR. DeepSeek V4 Pro 0813–DeepSeek Harness exceeds all 16 core configurations in Mean-of-4 on SWE-bench-45, Terminal-Bench 2, and VitaBench, but trails MiniMax–Codex on LOCA-Bench. A fixed native system is therefore neither an outcome frontier nor a valid-completion ceiling across task domains. Table 6 reports the complete comparisons.

The fixed references show that outcome and operational validity remain separate even for native systems. GPT-5.5–Codex is fully clean on VitaBench without leading in outcome. On Terminal-Bench 2, it leads every core configuration in outcome but leaves more than a quarter of attempts non-clean; on ClawMark, it leads in outcome but is less clean than five core configurations. DeepSeek V4 Pro 0813–DeepSeek Harness is clean on every SWE attempt, yet 53 remain unresolved. On Terminal-Bench 2, it also leads every core configuration in outcome, but its VCR trails most of them because more than a third of its runs terminate abnormally. On VitaBench, it leads every core configuration in outcome while remaining fully clean. On LOCA-Bench, both its VCR and

Mean-of-4 trail MiniMax–Codex. A native pairing is therefore neither a valid-completion ceiling nor a universal performance ceiling.

Table 6 compares each fixed reference with the full set of core configurations on exactly matched task support. For outcome or VCR z, the fixed-reference win rate is

$$R_r^z = \frac{100}{|\mathcal{B}_r|} \sum_{b \in \mathcal{B}_r} \frac{1}{|\mathcal{C}_b|} \sum_{c \in \mathcal{C}_b} \phi(z_{br}, z_{bc}),$$

where $\mathcal{C}_b$ is the set of evaluated core configurations and ties receive half credit. Joint Pareto dominance requires the reference to be no worse in both outcome and VCR and strictly better in at least one. Each benchmark receives equal weight. These quantities locate each complete system relative to the evaluated panel.

Table 6: Fixed references versus evaluated core configurations on matched tasks. Outcome: Mean-of-4 (continuous for ClawMark) or AgentSkillOS Task Score. W/T/L: configuration-level wins/ties/losses; WR: tie-aware reference win rate; Joint: percentage of configurations Pareto-dominated in both outcome and VCR. Summaries weight benchmarks equally: six for GPT-5.5–Codex; SWE-bench-45, Terminal-Bench 2, VitaBench, and LOCA-Bench for DeepSeek Harness.

Benchmark	$\lvert\mathcal{C}_b\rvert$	Native outcome	Outcome W/T/L	Outcome WR	Native VCR	VCR W/T/L	VCR WR	Joint
GPT-5.5–Codex								
SWE-bench-45	16	65.56	16/0/0	100.00	99.44	13/3/0	90.63	100.00
Terminal-Bench 2	16	60.39	16/0/0	100.00	72.47	10/0/6	62.50	62.50
VitaBench	16	46.56	11/0/5	68.75	100.00	4/12/0	62.50	68.75
LOCA-Bench	16	76.00	15/0/1	93.75	96.67	8/0/8	50.00	50.00
ClawMark	12	74.31	12/0/0	100.00	92.25	7/0/5	58.33	58.33
AgentSkillOS	12	97.57	12/0/0	100.00	100.00	12/0/0	100.00	100.00
Equal-benchmark mean	88	–	82/0/6	93.75	–	54/15/19	70.66	73.26
DeepSeek V4 Pro 0813–DeepSeek Harness								
SWE-bench-45	16	70.56	16/0/0	100.00	100.00	16/0/0	100.00	100.00
Terminal-Bench 2	16	46.91	16/0/0	100.00	62.92	4/0/12	25.00	25.00
VitaBench	16	54.13	16/0/0	100.00	100.00	4/12/0	62.50	100.00
LOCA-Bench	16	73.33	15/0/1	93.75	92.67	7/0/9	43.75	43.75
Equal-benchmark mean	64	–	63/0/1	98.44	–	31/12/21	57.81	67.19

On SWE-bench-45, DeepSeek V4 Pro 0813–DeepSeek Harness is five Mean-of-4 points above GPT-5.5–Codex, but the paired task-cluster bootstrap interval includes zero. On VitaBench, its 7.56-point advantage over GPT-5.5–Codex has a 95% interval of $[4.19, 10.94]$, while both systems have a 100% VCR. The outcome difference therefore remains after VCR saturates. On SWE-bench-45, the two systems also retain complementary task coverage: DeepSeek Harness uniquely solves four tasks and GPT-5.5–Codex uniquely solves three.

LOCA-Bench provides a contrasting native-system result. DeepSeek Harness trails both GPT-5.5–Codex and MiniMax–Codex in Mean-of-4 and VCR on this benchmark. Native adaptation is therefore useful as a fixed reference but does not impose a universal ceiling.

C Additional Outcome and Execution Analyses

C.1 Joint Outcomes and Pairing Residuals

Figure 5 reports all success/failure and clean/non-clean combinations on the four complete four-model benchmark panels. Figure 6 reports additive VCR residuals on the common three-model panel, separately by benchmark and in aggregate. The two views distinguish execution profiles from pairing-specific deviations beyond model and harness averages.

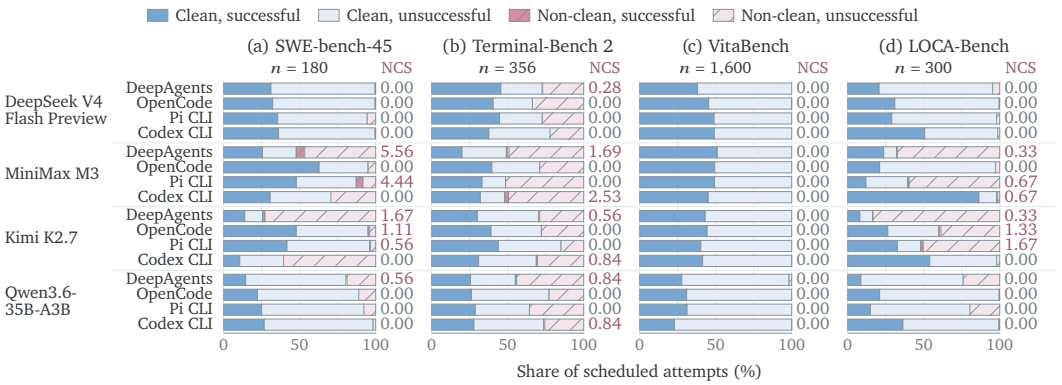

Figure 5: Joint outcome–validity shares for all 64 configurations (n: attempts per pairing). Bars sum to 100%; NCS gives non-clean success (%). Four repetitions per task; VitaBench weights domains equally. Fixed references are excluded.

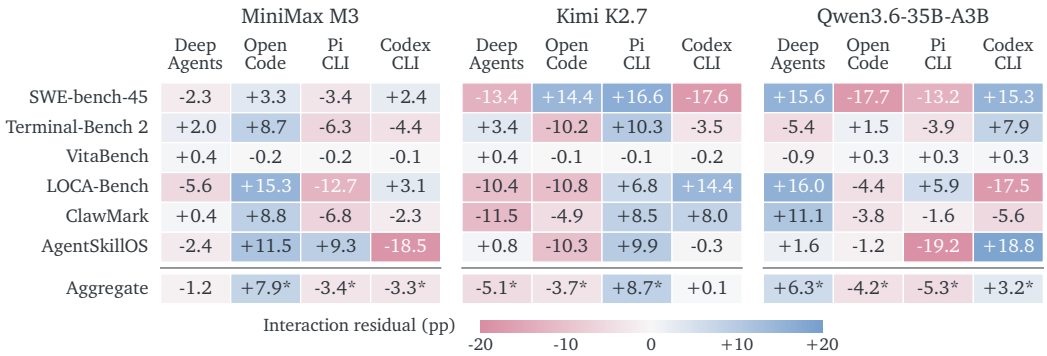

Figure 6: Additive VCR interaction residuals (pp) for the balanced three-model panel. Aggregate weights benchmarks equally; $*$ marks pointwise 95% task-cluster bootstrap intervals excluding zero. Benchmark cells report benchmark-specific point estimates.

C.2 Continuous-Score Decomposition

For ClawMark and the primary `skill_200` condition of AgentSkillOS, we retain the official continuous task score and separate the contributions from clean and non-clean attempts. Let z_i be the score of scheduled attempt i on a 0–100 scale, U_i its clean-run indicator, and n the number of scheduled attempts. The decomposition is

$$\text{Mean Score} = \frac{1}{n}\sum_{i=1}^{n} U_i z_i + \frac{1}{n}\sum_{i=1}^{n}(1-U_i)z_i. \tag{5}$$

Each component uses the full scheduled denominator, so their sum recovers the original Mean Score. Figure 7 reports the complete three-model, four-harness panels. ClawMark scores are multiplied by 100; AgentSkillOS retains its native 0–100 scale.

On ClawMark, Kimi–DeepAgents exceeds the fully clean Kimi–Codex by 46.7 Mean Score points, with nearly a third of its score contributed by non-clean attempts. On AgentSkillOS, MiniMax–Codex's mean score within clean attempts exceeds its overall Task Score by 58.4 points, but only just over a third of attempts are clean. These examples show why score contributions and clean-run coverage should be read together. Conditional group means are undefined when a group contains no attempts.

C.3 Execution-Stage Decomposition

VCR records normal termination with an evaluator-consumable final state. We complement it with an execution-stage decomposition of non-clean runs. We partition every non-clean run into one of two exclusive stages: a pre-normal loss when the harness does not terminate normally, and a post-normal

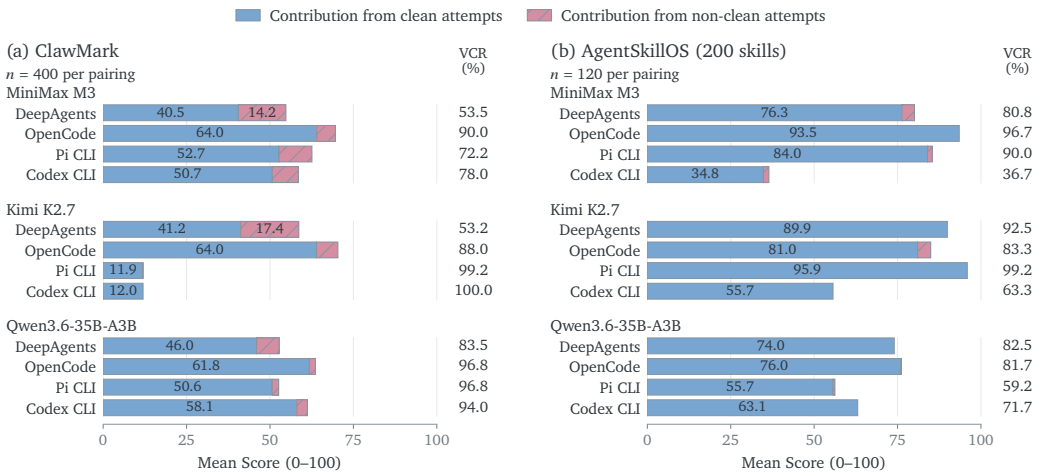

Figure 7: Clean/non-clean contributions summing to Mean Score, both divided by all scheduled attempts rather than group counts. Right column: VCR (%). ClawMark: 400 attempts per configuration; AgentSkillOS (`skill_200`): 120. Four repetitions per task; fixed references are excluded.

handoff loss when it terminates normally but fails to leave a valid state for evaluation. Together with VCR, the two stages sum to 100%. Figure 8 reports the decomposition both by harness and by benchmark on the balanced panel.

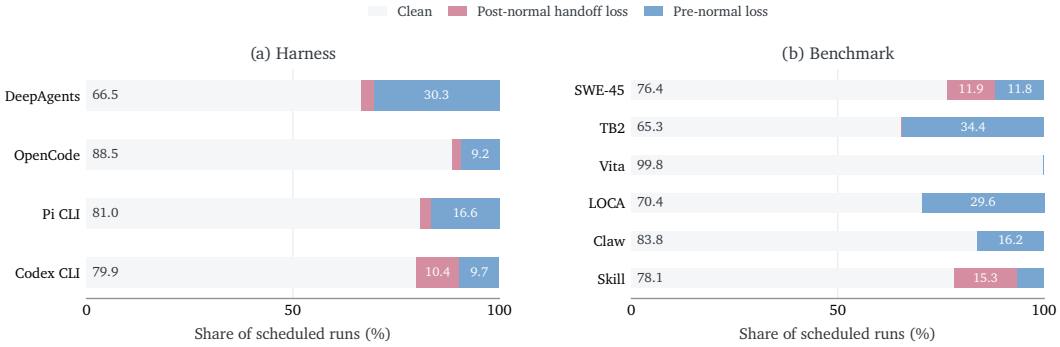

Figure 8: Exclusive execution stages on the balanced three-model panel: clean completion, pre-normal loss, and post-normal handoff loss sum to 100%. Macro-averages: (a) six benchmarks per harness; (b) four harnesses per benchmark. Fixed references are excluded.

The decomposition reveals distinct bottlenecks. Averaged equally across benchmarks, DeepAgents loses nearly ten times as many runs before normal termination as at post-normal handoff. OpenCode and Pi CLI show the same failure pattern at lower rates. Codex CLI differs: its losses are substantial and roughly balanced between these two stages. Similar Mean VCR values can therefore conceal failures in different stages of the interaction protocol.

Failure stages also vary by task. Panel (b) of Figure 8 shows that Terminal-Bench 2, LOCA-Bench, and ClawMark mainly expose failures before normal termination, whereas VitaBench is near ceiling. SWE-bench-45 and AgentSkillOS additionally expose evaluator-handoff losses, with AgentSkillOS losing 3.4 percentage points more attempts at this stage. Clean execution still does not imply task success: Kimi–Codex CLI is fully clean on ClawMark but scores poorly.

The decomposition maps directly to intervention targets. Pre-normal loss points to interface adherence, timeout handling, and recovery. Post-normal loss points to stopping, finalization, and submission validation. Low outcome after a clean run motivates trajectory inspection of planning, action realization, and task-specific support. These diagnostic associations connect execution stages with candidate intervention targets. Appendix A provides benchmark-specific execution details.

C.4 Case Study: Action Realization and Clean Completion

VCR measures process completion, while action realization captures the conversion of stated intent into executable actions. We focus on Kimi K2.7 on ClawMark, where native-response and control-flow evidence supports trajectory-level labels under four harnesses. Codex CLI and Pi CLI are almost perfectly clean yet score poorly. OpenCode outperforms Codex CLI by 58.4 Mean Score points despite a lower VCR; DeepAgents also substantially outperforms both nearly clean harnesses.

We use a scripted indicator to identify strict action-realization gaps. A strict *action-realization gap* requires prospective action intent in reasoning but no structured action or non-whitespace visible text in the same accepted response. Accepted responses define response boundaries; provider retries and no-tool context-compaction generations are handled separately. Native records and control flow classify all 400 attempts per harness. For 48 timed-out OpenCode attempts, the no-fallback control flow establishes the absence of strict gaps in the unrecorded stages.

Table 7: Strict Kimi K2.7 action-realization audit on ClawMark. Each harness has 400 attempts. Brackets show pointwise 95% task-cluster bootstrap intervals. For no-event cells, [0,0] reflects zero empirical bootstrap variation within the observed sample. Dashes denote undefined ratios.

Harness	Score Mean	VCR	Affected attempts % [95% CI]	Events per affected attempt
Codex CLI	11.96	100.00	6.75 [4.50, 9.25]	1.00
DeepAgents	58.65	53.25	0.00 [0.00, 0.00]	–
OpenCode	70.36	88.00	0.00 [0.00, 0.00]	–
Pi CLI	11.93	99.25	40.50 [36.25, 44.75]	1.32

Only Codex CLI and Pi CLI exhibit strict gaps. These gaps affect six times as many Pi attempts as Codex attempts and sometimes recur within a Pi attempt; Codex has exactly one event per affected attempt. The across-harness pattern coincides with the score split, but the within-harness association is small: affected attempts score less than 1.3 points below unaffected attempts in either harness.

Recovery policies differ: Codex retries malformed-action JSON, Pi restarts only after continuation-role errors, OpenCode has no external fallback, and DeepAgents runs a finalization pass. Yet OpenCode and DeepAgents both show zero events, while all but one Codex event and every Pi event have no linked later response; the remaining Codex retry is unresolved. The case compares recovery policies as deployed in the four harnesses and characterizes their associated trajectory patterns.

C.5 Repeated-Run Outcomes and Stress Conditions

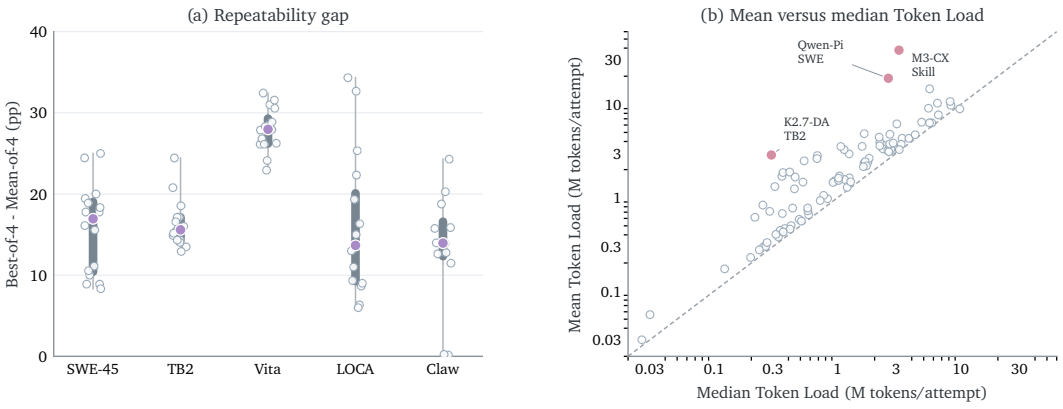

Figure 9: Repeated-run gaps and resource tails. (a) Best-of-4 minus Mean-of-4 for 76 core crossed configurations; thick segments: interquartile ranges; filled points: medians. (b) Mean versus median Token Load for 90 fully covered primary cells with both statistics; dashed line: equality. The three most skewed cells are highlighted.

Best-of-4 reports the best observed outcome for each task, whereas Mean-of-4 averages all four attempts. Panel (a) of Figure 9 shows positive gaps throughout the 76-cell repeated-run panel. VitaBench has the largest median gap, exceeding LOCA-Bench's smallest gap by about 14 percentage points. On SWE-bench-45, MiniMax–OpenCode's Best-of-4 exceeds its Mean-of-4 by 19.4 points. The fixed references show the same pattern on SWE-bench-45, Terminal-Bench 2, and VitaBench. Reliable harness operation therefore does not remove run-to-run variation in task success.

The stress analyses are also task-dependent. VitaBench's Cross environment is harder than every single-domain environment for every configuration. In AgentSkillOS, increasing the skill pool from 200 to 900 reduces GPT-5.5–Codex Task Score more than VCR. The larger skill pool therefore changes task performance more than the measured termination and handoff behavior for this reference.

D Cross-Model Fit and Validation

The Fit analysis uses the common three-model, four-harness panel across six benchmarks: 35,472 scheduled attempts in 72 configurations. The analysis uses the three models with common six-benchmark coverage and the primary AgentSkillOS condition; fixed references and stress conditions are reported separately. Every task retains all four scheduled repetitions. Scores are on a 0–1 scale before forming US: AgentSkillOS scores are divided by 100; the other benchmarks already use this scale. Each benchmark and each model receives equal weight.

Joint-score interpretation and aggregation. For a binary benchmark, J is the fraction of scheduled attempts that both succeed and achieve valid completion. For a continuous benchmark, it is the mean task-score contribution from valid completions. When VCR is positive, $J = \mathrm{VCR} \times \mathbb{E}[S \mid U = 1]$, using VCR as a fraction. The conditional mean is undefined when VCR is zero, while J remains zero. This is computed per attempt, rather than multiplying marginal mean outcome and VCR. The identity $\mathbb{E}[S] = \mathbb{E}[US] + \mathbb{E}[(1-U)S]$ preserves the official score and isolates the credit excluded by the joint criterion.

Fit is bounded between 0 and 100 and is unchanged by a positive rescaling of all scores within a model–benchmark block. With the other harnesses fixed, improving a harness's joint score cannot lower its own Fit. The geometric cross-model mean is no larger than the corresponding arithmetic mean, with equality when model-specific Fits are equal. A zero model-specific Fit makes the cross-model index zero. Fit uses the best-score reference within the stated candidate-harness panel, so reference values change with that panel.

Table 8: Fit on the balanced six-benchmark panel (0–100). Brackets give pointwise 95% paired task-cluster bootstrap intervals.

Harness	MiniMax M3	Kimi K2.7	Qwen3.6	Cross-model
OpenCode	86.86 [85.11, 88.50]	87.06 [83.02, 90.94]	88.55 [80.84, 93.36]	87.48 [84.06, 89.99]
Pi CLI	73.73 [69.87, 77.73]	76.16 [71.99, 80.27]	81.70 [74.88, 84.98]	77.13 [73.49, 79.62]
Codex CLI	72.37 [68.59, 76.61]	60.45 [56.84, 63.91]	91.21 [85.55, 93.44]	73.62 [70.54, 75.82]
DeepAgents	60.56 [56.35, 64.62]	61.29 [57.11, 65.48]	71.29 [63.99, 76.22]	64.20 [60.57, 66.94]

Component comparisons. The outcome-only variant uses the same normalization and geometric aggregation after replacing US by S. The arithmetic variant retains the joint score and normalization but averages model-specific Fits arithmetically. Table 3 also reports the existing outcome and VCR win rates. Joint and official scores are equal in 37 of the 72 benchmark-level cells; the remaining 35 lose some credit under the valid-completion condition. On SWE-bench-45, MiniMax–DeepAgents has official and joint scores of 31.11 and 25.56, respectively, while both scores for MiniMax–Codex are 30.56. On ClawMark, Kimi–DeepAgents loses 17.43 score points, the largest cell-level difference. These comparisons retain all scheduled attempts in the denominator.

The geometric aggregation lowers Codex CLI's index by 1.06 points relative to the arithmetic variant, the largest such reduction among the four harnesses. Both component substitutions preserve the overall ordering; the outcome-only variant also preserves each model's harness ordering. These comparisons show how Fit decomposes joint performance and summarizes its balance across models.

Uncertainty and panel sensitivity. The 2,000 paired task-cluster bootstrap replicates preserve all four repetitions and every model–harness observation for a sampled task. Resampling occurs within benchmarks, with separate strata for VitaBench's four environments. Each replicate recomputes joint scores, best-harness references, model-specific Fits, and the cross-model geometric mean. No reference denominator is zero in the point estimates or replicates. Intervals are pointwise percentiles and describe task variation conditional on the evaluated model and benchmark panel. The Pi–Codex index difference is 3.50 points (95% paired interval: $[-0.26, 7.05]$). Leave-one-out summaries recompute the metric on the retained panel.

Table 9: Cross-model Fit after omitting one model or benchmark. Reference scores and aggregation are recomputed on the retained panel. Rankings use unrounded estimates. DA: DeepAgents; OC: OpenCode; Pi: Pi CLI; CX: Codex CLI.

Omitted		DA	OC	Pi	CX	Order
Model	MiniMax	66.10	87.80	78.88	74.26	$OC > Pi > CX > DA$
Model	Kimi	65.71	87.70	77.61	81.25	$OC > CX > Pi > DA$
Model	Qwen	60.92	86.96	74.94	66.14	$OC > Pi > CX > DA$
Benchmark	SWE-45	68.86	86.06	75.45	77.75	$OC > CX > Pi > DA$
Benchmark	TB2	63.29	86.29	73.66	71.91	$OC > Pi > CX > DA$
Benchmark	Vita	57.82	85.26	73.40	70.69	$OC > Pi > CX > DA$
Benchmark	LOCA	72.63	96.23	84.82	67.98	$OC > Pi > DA > CX$
Benchmark	Claw	63.60	84.98	80.19	76.23	$OC > Pi > CX > DA$
Benchmark	Skill	58.87	86.01	74.88	76.59	$OC > CX > Pi > DA$

Removing any one of the four harnesses preserves the relative ordering among the survivors. Their scores still change when the omitted harness supplied a best-score reference, illustrating the metric's dependence on the candidate set. Removing models or benchmarks can additionally change the intermediate ordering, as shown in Table 9.

Leave-one-model-out selection diagnostic. For each omitted model, select the harness with the highest aggregate score on the two remaining models, then assess its Fit on the omitted model. Outcome Win Rate and VCR Win Rate average their model-conditioned rates over the training models; the outcome-only and joint Fit variants use the same training-model geometric aggregation. The omitted model contributes no data to selection, but its full harness panel defines the evaluation reference. All four rules select OpenCode in all three folds. It is the best single harness for MiniMax and Kimi, while its Qwen Fit is 2.66 points below Codex CLI. This retrospective diagnostic tests selection consistency across the observed models.

Table 10: Leave-one-model-out selection diagnostic. Fit, outcome-only Fit, Outcome Win Rate, and VCR Win Rate all select OpenCode using the two remaining models. Regret is the difference, in Fit points, between the held-out model's best single-harness Fit and the selected Fit.

Held-out model	Selected Fit	Best harness	Best Fit	Regret
MiniMax M3	86.86	OpenCode	86.86	0.00
Kimi K2.7	87.06	OpenCode	87.06	0.00
Qwen3.6	88.55	Codex CLI	91.21	2.66

E VCR Summaries and Sensitivity

To complement mean summaries, we report the type-7 25th percentile and minimum of VCR. We also obtain model-side VCR win rates by comparing models within benchmark–harness blocks.

The type-7 lower quartile interpolates between adjacent observations in the fixed evaluation panel. As a discrete sensitivity, we instead take the nearest-rank empirical quartile—the $\lceil 0.25n \rceil$th ordered observation. This change preserves both orderings: OpenCode remains the leading harness and Qwen the leading model (Table 11). We retain the type-7 statistic as the interpolated lower-quartile summary and report the observed minimum to represent worst-configuration VCR.

Table 11: Nearest-rank lower quartiles of VCR (%). Harness and model orderings match the type-7 summaries.

Harness	Lower quartile	Model	Lower quartile
DeepAgents	49.16	MiniMax M3	48.60
OpenCode	81.67	Kimi K2.7	60.00
Pi CLI	64.33	Qwen3.6-35B-A3B	76.00
Codex CLI	68.82		

F Benchmark-Specific Results

The following sections retain the complete benchmark-level tables and benchmark-specific analyses underlying the cross-benchmark findings in the main text.

F.1 SWE-bench-45

Table 12 reports the complete SWE-bench-45 results. The gap between Best-of-4 and Mean-of-4 is substantial in several configurations. MiniMax M3 with OpenCode, for example, reaches the highest Best-of-4 and Mean-of-4 in its model block, but the 19.4-point gap still indicates that some long-horizon issues are solved in at least one trial but not reliably across all four executions. Harness rankings are model dependent: Codex CLI attains the highest Mean-of-4 for DeepSeek V4 Flash Preview and Qwen3.6-35B-A3B, and OpenCode for MiniMax M3 and Kimi K2.7.

SWE-bench-45 operational validity. The four-model panel contains 2,880 scheduled SWE executions. The fixed GPT-5.5–Codex and DeepSeek V4 Pro 0813–DeepSeek Harness references each contribute a separate 180-run panel on the same 45 tasks. Here a valid submission is a non-empty patch accepted by the evaluator's patch-application stage; correctness is excluded from operational validity. Table 13 reports the single-benchmark VCR matrix and its within-SWE model and harness summaries.

DeepSeek combines near-ceiling operational validity across harnesses with a best Mean-of-4 success rate of 36.11%. Among the other models, OpenCode and Pi CLI produce clean evaluator-ready completions in more than 87% of runs for every model. Kimi has the lowest VCR under DeepAgents. Codex CLI shows the opposite model dependence: its VCR is 58.9 percentage points higher for Qwen than for Kimi. Twenty-five runs that ultimately solve the issue are not counted as valid completions because they terminate abnormally, illustrating why evaluator handoff and task success are distinct. Both fixed references combine near-ceiling VCR with substantial Best-of-4–Mean-of-4 gaps, separating operational validity from repeatable issue resolution.

The trajectory evidence localizes mechanisms consistent with these residuals. DeepAgents records nearly twice as many `RuntimeError` failures for Kimi as for MiniMax on the same scheduled support, while Qwen has none. Codex CLI terminates normally in all Qwen runs and produces an evaluator-ready patch in 106 more executions than for Kimi out of the same 180 attempts. These SWE-specific residuals guide trajectory inspection and depend on the comparison panel. Their magnitudes attenuate in the matched three-model calculation in Figure 6; including DeepSeek changes the centering baseline and yields the four-model values reported here.

F.2 Terminal-Bench 2

Across the full 89 tasks in Terminal-Bench 2, performance varies substantially across model–harness configurations. Pi CLI has the strongest four-model macro-average, exceeding runner-up OpenCode by 1.3 percentage points in Mean-of-4. Its largest model-specific advantage is on Kimi, where it

Table 12: SWE-bench-45 Best-of-4 and Mean-of-4 success rates (%; Appendix G). Bold/underlined: best/second-best within each core model, including ties. Fixed references are excluded from highlighting.

Model	Harness	Best	Mean
DeepSeek V4 Flash Preview	DeepAgents	40.00	31.11
	OpenCode	42.22	32.22
	Pi CLI	**46.67**	<u>35.56</u>
	Codex CLI	**46.67**	**36.11**
MiniMax M3	DeepAgents	55.56	31.11
	OpenCode	**82.22**	**62.78**
	Pi CLI	<u>71.11</u>	<u>52.22</u>
	Codex CLI	55.56	30.56
Kimi K2.7	DeepAgents	31.11	15.56
	OpenCode	**66.67**	**48.89**
	Pi CLI	<u>51.11</u>	<u>42.22</u>
	Codex CLI	28.89	10.56
Qwen3.6-35B-A3B	DeepAgents	31.11	15.00
	OpenCode	<u>42.22</u>	22.22
	Pi CLI	33.33	<u>25.00</u>
	Codex CLI	**44.44**	**26.67**
GPT-5.5 xhigh + Codex CLI		86.67	65.56
DeepSeek V4 Pro 0813 + DeepSeek Harness		88.89	70.56

Table 13: SWE-bench-45 VCR (%). A: four-model matrix and fixed references. B: core-panel harness/model pairwise win rates, means, type-7 25th percentiles, and minima. These single-benchmark summaries differ from the balanced three-model, six-benchmark summaries in Table 5.

Panel A: VCR matrix

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI	DeepSeek Harness
DeepSeek V4 Flash Preview	99.44	99.44	94.44	99.44	–
MiniMax M3	47.78	95.00	87.22	70.56	–
Kimi K2.7	25.56	95.00	96.11	39.44	–
Qwen3.6-35B-A3B	80.56	88.89	92.22	98.33	–
GPT-5.5 xhigh	–	–	–	99.44	–
DeepSeek V4 Pro 0813	–	–	–	–	100.00

Panel B: single-benchmark pairwise and magnitude summaries

Comparison	Name	Pairwise win rate	Mean	25th pct.	Minimum
Harness	DeepAgents	16.67	63.33	42.22	25.56
	OpenCode	66.67	94.58	93.47	88.89
	Pi CLI	58.33	92.50	90.97	87.22
	Codex CLI	58.33	76.94	62.78	39.44
Model	DeepSeek V4 Flash Preview	91.67	98.19	98.19	94.44
	MiniMax M3	29.17	75.14	64.86	47.78
	Kimi K2.7	37.50	64.03	35.97	25.56
	Qwen3.6-35B-A3B	41.67	90.00	86.81	80.56

leads both metrics and exceeds OpenCode by 4.8 points in Mean-of-4. For Qwen, Codex CLI leads Best-of-4 and ties Pi CLI on Mean-of-4. The ordering changes for the other models: DeepAgents leads DeepSeek Mean-of-4 and ties Pi CLI on Best-of-4, whereas OpenCode leads both MiniMax M3 metrics. The leading harness varies across models.

Table 14: SWE-bench-45 additive VCR interaction residual (percentage points), centered over the four-model SWE panel.

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
DeepSeek V4 Flash Preview	19.76	-11.49	-14.41	6.15
MiniMax M3	-8.85	7.12	1.42	0.31
Kimi K2.7	-19.97	18.23	21.42	-19.69
Qwen3.6-35B-A3B	9.06	-13.85	-8.44	13.23

Table 15: Terminal-Bench 2 results on all 89 tasks: four models, four harnesses, and fixed references. Best/Mean: Best-of-4/Mean-of-4.

Model	DeepAgents		OpenCode		PI		Codex	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean
Qwen3.6-35BA3B	40.4	26.4	40.4	26.1	43.8	**28.7**	**47.2**	**28.7**
DeepSeek	**61.8**	**45.8**	53.9	40.4	**61.8**	44.7	50.6	37.6
MiniMax-M3	46.1	21.6	**55.1**	**39.6**	53.9	33.1	51.7	34.6
Kimi-K2.7	44.9	30.6	53.9	39.0	**59.6**	**43.8**	48.3	31.7

Fixed system reference	Best-of-4 (%)	Mean-of-4 (%)
GPT-5.5 xhigh–Codex CLI	76.40	60.39
DeepSeek V4 Pro 0813–DeepSeek Harness	62.92	46.91

From the model perspective, DeepSeek V4 Flash Preview remains strongest overall, while Qwen obtains the lowest averages. Harness choice still affects every model: for MiniMax, OpenCode exceeds DeepAgents by 18.0 percentage points in Mean-of-4. The large Best-of-4–Mean-of-4 gaps distinguish occasional task success from consistent performance across repeated executions.

The harness ranking is model dependent: Codex CLI leads Qwen Best-of-4 and ties Pi CLI on Mean-of-4; Pi CLI leads Kimi; DeepAgents leads DeepSeek Mean-of-4 and ties Pi CLI on Best-of-4; OpenCode leads MiniMax. These reversals characterize model-dependent performance across full system configurations that combine prompts, adapters, action-schema formatting, context management, and runtime settings.

Both fixed references exceed every core configuration in outcome. GPT-5.5–Codex leads DeepSeek V4 Pro 0813–DeepSeek Harness by 13.5 percentage points in both Best-of-4 and Mean-of-4. Neither system is fully clean. More than a third of DeepSeek Harness runs terminate abnormally, while its final-verifier handoff rate is near ceiling; its valid-completion failures therefore arise almost entirely from abnormal termination. Seven non-normal runs still receive full task credit. Terminal-Bench 2 therefore provides a direct example in which official performance hides execution instability.

F.3 VITABENCH

Table 16 reports Best-of-4 and Mean-of-4 under strict tool permissions. Cross-domain interaction is the most difficult setting under both metrics for every model–harness configuration. In the core panel, the best Cross configuration falls 20.8 percentage points below the best single-domain configuration in Mean-of-4; the Best-of-4 gap is 13 points. Harness effects remain model dependent. By Total Mean-of-4, Codex CLI leads for DeepSeek, DeepAgents for MiniMax, OpenCode for Kimi, and Pi CLI for Qwen. By Total Best-of-4, Codex CLI leads for DeepSeek, MiniMax, and Kimi, while Pi CLI leads for Qwen.

Across the core configurations, the gap between unrounded Total Best-of-4 and Total Mean-of-4 is roughly 23–32 percentage points. Thus, the configurations solve a substantially broader set of tasks at least once than they solve reliably across repeated executions. The fixed DeepSeek V4 Pro 0813–DeepSeek Harness reference extends this trend: its Best-of-4 exceeds its Mean-of-4 by 25.1 points, although the latter still exceeds every core configuration and GPT-5.5–Codex. Both references are excluded from the within-model formatting.

Table 16: VitaBench Best-of-4/Mean-of-4 success rates (%) under strict tool permissions: 100 tasks and four independent trials (400 attempts) per environment (Appendix G). Cross combines Delivery, In-store, and OTA; Total averages the four environments equally. Bold/underlined: best/second-best per core model and metric, including ties. Fixed references are excluded from highlighting.

Model	Harness	Delivery		In-store		OTA		Cross		Total	
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
MiniMax M3	DeepAgents	81.00	**59.50**	83.00	58.25	76.00	47.00	69.00	39.25	77.25	**51.00**
	OpenCode	83.00	59.00	84.00	56.75	76.00	49.00	59.00	32.75	75.50	49.38
	Pi CLI	**86.00**	59.25	**87.00**	**58.50**	**77.00**	**49.25**	59.00	30.00	77.25	49.25
	Codex CLI	85.00	53.25	82.00	48.50	67.00	35.25	**76.00**	**43.25**	**77.50**	45.06
Kimi K2.7	DeepAgents	75.00	51.50	78.00	52.25	69.00	**47.00**	42.00	21.50	66.00	43.06
	OpenCode	76.00	**54.50**	83.00	**56.00**	62.00	42.25	**53.00**	**24.75**	68.50	**44.38**
	Pi CLI	75.00	48.75	84.00	52.25	67.00	40.75	49.00	19.25	68.75	40.25
	Codex CLI	**79.00**	50.75	**89.00**	50.50	**72.00**	44.00	48.00	20.50	**72.00**	41.44
Qwen3.6-35B-A3B	DeepAgents	74.00	39.75	**75.00**	35.25	**60.00**	26.00	28.00	9.75	59.25	27.69
	OpenCode	69.00	41.25	72.00	37.25	55.00	**29.50**	**39.00**	**15.50**	58.75	30.88
	Pi CLI	**76.00**	**44.75**	73.00	**38.00**	53.00	27.75	36.00	14.25	**59.50**	**31.19**
	Codex CLI	68.00	35.50	74.00	31.25	42.00	17.25	23.00	7.50	51.75	22.88
DeepSeek V4 Flash Preview	DeepAgents	78.00	46.75	80.00	48.25	73.00	39.25	45.00	17.75	69.00	38.00
	OpenCode	83.00	58.75	86.00	52.00	65.00	42.00	52.00	28.75	71.50	45.38
	Pi CLI	85.00	60.50	**87.00**	**53.75**	70.00	**46.75**	61.00	**34.75**	75.75	48.94
	Codex CLI	**86.00**	**64.00**	86.00	52.75	**74.00**	44.75	**62.00**	**34.75**	**77.00**	**49.06**
GPT-5.5 xhigh + Codex CLI		81.00	58.75	80.00	51.50	71.00	48.75	44.00	27.25	69.00	46.56
DeepSeek V4 Pro 0813 + DeepSeek Harness		87.00	61.00	90.00	62.75	73.00	52.00	67.00	40.75	79.25	54.13

VitaBench operational validity. The four-model strict panel contains 25,600 scheduled attempts. The DeepSeek Harness reference contributes a separate 1,600-run panel on the same task–trial support and reaches a 100% VCR. Table 17 reports the complete single-benchmark process profile.

VitaBench VCR is near ceiling: every core configuration has a VCR of at least 98%, and the within-benchmark additive residuals lie within one percentage point of zero. Outcome remains far from saturated: Total Mean-of-4 differs by 28.1 points between the strongest and weakest core configurations, and the leading harness still changes with the model. VitaBench therefore isolates a regime in which model–harness outcome differences persist even when almost every run executes and hands off successfully. Both fixed references are fully clean, yet DeepSeek Harness exceeds GPT-5.5–Codex by 7.6 points in Mean-of-4, reinforcing the same separation.

Resource use also shows a model-conditioned difference. For DeepSeek, Pi CLI uses less than half as many tokens per attempt as Codex CLI, while trailing it by less than 0.2 percentage points in Mean-of-4. These fully covered cells use the same accounting rule as the cross-benchmark resource analysis in Section 6.1.

Table 17: VitaBench process metrics under strict permissions. A: VCR (%) and model/harness summaries. B: four-model interaction residuals (pp). C: harness-level normal termination, valid handoff, and VCR (%). D: fixed references.

Panel A: VCRs with model and harness summaries

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI	Mean	25th pct.	Minimum	Model-side win rate
DeepSeek V4 Flash Preview	100.00	99.94	100.00	100.00	99.98	99.98	99.94	50.00
MiniMax M3	100.00	100.00	100.00	100.00	100.00	100.00	100.00	66.67
Kimi K2.7	100.00	100.00	100.00	99.94	99.98	99.98	99.94	50.00
Qwen3.6-35B-A3B	98.25	100.00	100.00	99.94	99.55	99.52	98.25	33.33
Harness mean	99.56	99.98	100.00	99.97	–	–	–	–
Harness 25th pct.	99.56	99.98	100.00	99.94	–	–	–	–
Harness minimum	98.25	99.94	100.00	99.94	–	–	–	–
VCR win rate	45.83	50.00	66.67	37.50	–	–	–	–

Panel B: additive interaction residual

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
DeepSeek V4 Flash Preview	+0.33	-0.15	-0.11	-0.07
MiniMax M3	+0.32	-0.11	-0.12	-0.09
Kimi K2.7	+0.33	-0.09	-0.11	-0.14
Qwen3.6-35B-A3B	-0.98	+0.35	+0.33	+0.30

Panel C: harness-level termination and handoff components

Harness	Normal exit	Valid handoff	VCR
DeepAgents	99.56	100.00	99.56
OpenCode	99.98	100.00	99.98
Pi CLI	100.00	100.00	100.00
Codex CLI	99.97	100.00	99.97

Panel D: fixed system references

Configuration	Normal exit	Valid handoff	VCR
GPT-5.5–Codex	100.00	100.00	100.00
DeepSeek V4 Pro 0813–DeepSeek Harness	100.00	100.00	100.00

F.4 LOCA-BENCH

Main results. Table 18 reports the primary outcomes. Codex CLI is the best harness within every model block. Its advantage is especially large for MiniMax M3: Codex CLI exceeds runner-up DeepAgents by 63 percentage points in Mean-of-4. For each remaining model, its lead over the next-best harness is roughly 15–20 points. MiniMax–Codex exceeds GPT-5.5–Codex by 11 Mean-of-4 points and exceeds both fixed references in Best-of-4.

Table 18: LOCA-Bench 256K Best-of-4/Mean-of-4 success rates (%) for the four-model, four-harness panel and fixed references. Bold marks the best harness per core model and metric; references are excluded.

Model	DeepAgents		OpenCode		Pi CLI		Codex CLI	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean
DeepSeek V4 Flash Preview	40.00	20.67	53.33	31.00	42.67	29.00	**76.00**	**50.67**
Qwen 3.6–35B–A3B	14.67	8.67	34.67	21.00	28.00	15.00	**70.67**	**36.33**
MiniMax M3	33.33	24.00	32.00	21.00	21.33	12.67	**93.33**	**87.00**
Kimi K2.7	17.33	8.33	42.67	27.67	50.67	34.33	**86.67**	**54.00**

Fixed system reference	Best-of-4 (%)	Mean-of-4 (%)
GPT-5.5 xhigh–Codex CLI	90.67	76.00
DeepSeek V4 Pro 0813–DeepSeek Harness	90.67	73.33

Operational validity. The four-model panel contains $4 \times 4 \times 75 \times 4 = 4{,}800$ executions, and each fixed system reference contributes 300 runs on the same 75 tasks. The normal-exit condition requires `status=success`. The valid-handoff condition requires a finite final-state evaluator result, including a result recorded before a subsequently abnormal backend exit. A run is clean only when both conditions hold. Table 19 reports the resulting four-model panel.

Table 19: LOCA-Bench VCR (%). A: four-model matrix; B: fixed references; C: core-panel harness/model pairwise win rates, means, type-7 25th percentiles, and minima.

Panel A: VCR matrix

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
DeepSeek V4 Flash Preview	95.33	99.67	98.00	98.67
Qwen3.6-35B-A3B	76.00	99.67	80.33	99.33
MiniMax M3	32.33	97.33	39.67	98.00
Kimi K2.7	16.33	60.00	48.00	98.00

Panel B: fixed system references

System	Normal exit	Valid output	VCR
GPT-5.5 xhigh–Codex CLI	96.67	100.00	96.67
DeepSeek V4 Pro 0813–DeepSeek Harness	92.67	100.00	92.67

Panel C: single-benchmark pairwise and magnitude summaries

Comparison	Name	Pairwise win rate	Mean	25th pct.	Minimum
Harness	DeepAgents	0.00	55.00	28.33	16.33
	OpenCode	83.33	89.17	88.00	60.00
	Pi CLI	33.33	66.50	45.92	39.67
	Codex CLI	83.33	98.50	98.00	98.00
Model	DeepSeek V4 Flash Preview	87.50	97.92	97.33	95.33
	Qwen3.6-35B-A3B	79.17	88.83	79.25	76.00
	MiniMax M3	20.83	66.83	37.83	32.33
	Kimi K2.7	12.50	55.58	40.08	16.33

The interaction is substantially larger than a harness-wide average suggests. For the same DeepAgents interface, DeepSeek's VCR exceeds Kimi's by 79 percentage points. For MiniMax, Codex CLI exceeds DeepAgents by 65.7 points. DeepAgents' valid-state rate equals its VCR in all four model configurations, while valid-state rates for the other harnesses are at or near ceiling. Thus, the large LOCA losses are primarily failures of normal execution rather than failures of the final-state evaluator after a normal stop. The same point is visible in the four-model double-centered residuals in Table 20. Both native references have lower VCRs than several core configurations, and their non-normal runs retain evaluator records with zero task scores.

Harness and model effects. The marginal means in Table 21 show that harness choice has a larger effect than model choice in this setting. The spread between harness marginal means is 2.6 times the spread between model marginal means for Mean-of-4, and 3.5 times for Best-of-4. A sum-of-squares decomposition likewise assigns most aggregate configuration variation to the harness main effect, followed by the interaction and then the model main effect (Table 21, lower panel). These shares describe the distribution of variation among the aggregated configuration means.

The model–harness interaction produces ranking reversals. MiniMax is best under DeepAgents and Codex CLI, DeepSeek is best under OpenCode, and Kimi is best under Pi CLI. Model rankings are therefore conditional on the execution framework.

Token efficiency and runtime evidence. For DeepSeek, Codex CLI uses about 86% fewer tokens per attempt than Pi CLI while increasing Mean-of-4 by 21.7 percentage points. The higher success rate also widens its resource advantage: tokens per success are about 92% lower.

Trajectory status makes the system-level interpretation more concrete. We show a runtime-error comparison on Kimi in Table 22. Codex has 114 fewer runtime errors than runner-up OpenCode across the same 300 attempts, and no timeout-like failures. Some non-Codex runs have an evaluator-correct environment state despite an error status, showing that the score can mix task failure with unreliable process termination or result write-back. DeepAgents additionally exhibits recursion-limit and partial-state recovery events; OpenCode errors in this slice are uniformly timeout-like;

Table 20: LOCA-Bench additive VCR interaction residual (percentage points), centered over the four-model panel.

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
DeepSeek V4 Flash Preview	+19.71	-10.13	+10.88	-20.46
Qwen3.6-35B-A3B	+9.46	-1.04	+2.29	-10.71
MiniMax M3	-12.21	+18.63	-16.38	+9.96
Kimi K2.7	-16.96	-7.46	+3.21	+21.21

Table 21: LOCA-Bench marginal outcomes and descriptive shares of configuration variation (%). Marginal means average over four models or four harnesses.

Marginal outcomes

Grouping	Level	Best-of-4 (%)	Mean-of-4 (%)
Harness	DeepAgents	26.33	15.42
	OpenCode	40.67	25.17
	Pi CLI	35.67	22.75
	Codex CLI	**81.67**	**57.00**
Model	DeepSeek V4 Flash Preview	**53.00**	32.83
	Qwen 3.6–35B–A3B	37.00	20.25
	MiniMax M3	45.00	**36.17**
	Kimi K2.7	49.33	31.08

Shares of configuration variation

Outcome	Harness	Model	Interaction
Mean-of-4	67.2	9.4	23.4
Best-of-4	81.9	6.5	11.6

Pi CLI shows timeout-like exits and broken pipes. These observations characterize LOCA-Bench performance through context organization, tool protocol, and runtime reliability.

Under the 256K, multi-tool, exact-state regime, variation among harness marginal means exceeds variation among model marginal means. The trajectory analysis characterizes differences in context selection, tool-result management, and stopping behavior across the evaluated systems.

F.5 CLAWMARK

Table 23 reports ClawMark results for three models across four harnesses and the fixed native reference. OpenCode achieves the strongest aggregate core-panel results, with all three models obtaining high scores and no model-specific collapse. M3 shows the most balanced high-score profile across harnesses, remaining competitive under OpenCode, DeepAgents, Pi, and Codex. K2 exhibits the sharpest harness dependence: it achieves the highest Mean-of-4 point estimate under OpenCode and remains competitive under DeepAgents, but collapses under Pi and Codex. Qwen shows broad compatibility across four harnesses, although its best results do not reach the strongest M3 or K2 configurations. The GPT-5.5–Codex reference exceeds the strongest core configuration, Kimi–OpenCode, by 4.0 Mean-of-4 score points and also leads in Best-of-4. These results show that harness choice directly affects how model capabilities are expressed during task execution, motivating interpretation at the model–harness level.

Appendix C.4 examines K2's outcome–validity discrepancy using strict response-level annotations with fully observable cross-harness denominators and task-level uncertainty.

Beyond this compatibility issue, the harnesses differ in how they sustain long-horizon execution. OpenCode generally reaches high scores through relatively compact trajectories, translating model decisions into executable actions and state updates with fewer execution steps. DeepAgents generally produces longer trajectories with substantially more recorded execution steps. This extended execution increases exposure to turn-level timeouts and external service or transport failures. Some interrupted trajectories still receive substantial rubric credit, indicating that meaningful progress was

Table 22: Runtime-error evidence for Kimi K2.7 (300 attempts per harness).

Harness	Runtime errors	Timeout-like failures	Main observed pattern
Codex CLI	6	0	Most runs terminate normally.
Pi CLI	156	154	Timeouts, and `BrokenPipeError`; some error runs still have a correct final state.
OpenCode	120	120	Timeout termination during long tasks.
DeepAgents	251	235	Long timeouts, graph recursion limits, forced completion, and partial-state recovery.

made before termination. Pi and Codex remain effective for Qwen and M3, but their performance depends more strongly on whether the model consistently follows the expected structured-action protocol.

Table 23: ClawMark Best-of-4/Mean-of-4 results (%). Score uses continuous evaluator scores; Full Success requires a run score of 1. Bold/underlined: best/second-best per core model and metric, including ties. The fixed native reference is excluded from highlighting.

Model	Harness	Score		Full Success	
		Best	Mean	Best	Mean
MiniMax M3	DeepAgents	75.06	54.78	21.00	11.00
	OpenCode	**82.39**	**69.62**	**28.00**	<u>12.75</u>
	Pi CLI	<u>78.34</u>	<u>62.58</u>	**28.00**	**14.50**
	Codex CLI	77.25	58.49	<u>22.00</u>	11.75
Kimi K2.7	DeepAgents	<u>72.56</u>	<u>58.65</u>	<u>20.00</u>	<u>8.25</u>
	OpenCode	**81.83**	**70.36**	**22.00**	**10.25**
	Pi CLI	12.08	11.93	0.00	0.00
	Codex CLI	12.20	11.96	0.00	0.00
Qwen3.6-35B-A3B	DeepAgents	68.58	52.71	10.00	4.50
	OpenCode	<u>76.25</u>	**63.62**	<u>11.00</u>	<u>4.75</u>
	Pi CLI	**76.91**	52.61	**20.00**	**6.50**
	Codex CLI	75.19	<u>61.21</u>	10.00	<u>4.75</u>
GPT-5.5 xhigh + Codex CLI		82.54	74.31	28.00	16.25

ClawMark operational validity. The core panel contains 4,800 scheduled attempts, and the fixed native reference contributes another 400 attempts on the same 100 tasks. The valid-handoff rate is 100 in every configuration, so VCR is determined by normal termination. Table 24 reports the resulting matrix and single-benchmark interaction residuals.

Table 24: ClawMark process metrics. A: core-panel VCR (%) with model/harness summaries and a separate fixed reference. B: core-panel additive interaction residuals (pp).

Panel A: VCRs with model and harness summaries

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI	Mean	25th pct.	Minimum	Model-side win rate
MiniMax M3	53.50	90.00	72.25	78.00	73.44	67.56	53.50	25.00
Kimi K2.7	53.25	88.00	99.25	100.00	85.13	79.31	53.25	50.00
Qwen3.6-35B-A3B	83.50	96.75	96.75	94.00	92.75	91.38	83.50	75.00
GPT-5.5 xhigh (native)	–	–	–	92.25	–	–	–	–
Harness mean	63.42	91.58	89.42	90.67	–	–	–	–
Harness 25th pct.	53.38	89.00	84.50	86.00	–	–	–	–
Harness minimum	53.25	88.00	72.25	78.00	–	–	–	–
VCR win rate	0.00	72.22	61.11	66.67	–	–	–	–

Panel B: additive interaction residual

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
MiniMax M3	+0.42	+8.75	-6.83	-2.33
Kimi K2.7	-11.52	-4.94	+8.48	+7.98
Qwen3.6-35B-A3B	+11.10	-3.81	-1.65	-5.65

For Kimi, Pi CLI and Codex CLI combine near-ceiling VCR with low Mean-of-4 scores, separating valid completion from task-solving performance. For MiniMax, OpenCode leads VCR comparisons and exceeds its additive baseline, whereas for Kimi the positive residuals move to Pi CLI and Codex CLI. Qwen has the highest model-side VCR win rate, but its largest positive residual is with DeepAgents. These VCR patterns vary with both the harness and the model even within one long-horizon workplace benchmark.

The native reference leads Score Mean-of-4 while ranking below five core configurations in VCR. All 31 non-normal runs retain evaluator scores, including eight with full success, showing that outcome and termination metrics capture distinct aspects of execution.

Among configurations that sustain task execution, OpenCode combines the strongest aggregate outcomes with comparatively compact trajectories. For MiniMax M3, it uses about 44% fewer tokens

per attempt than DeepAgents. Kimi K2.7 under Pi CLI and Codex CLI combines low Token Load with low task scores, illustrating the value of interpreting resource use together with outcomes.

F.6 AGENTSKILLOS

Table 25 reports the official evaluator score for each configuration. The larger skill pool changes the ordering rather than shifting all configurations uniformly. OpenCode leads MiniMax and Qwen under the primary 200-skill condition, whereas Pi CLI leads both models under the 900-skill stress condition. Pi CLI is strongest for Kimi in both conditions. Skill-pool scale therefore interacts with both the model and the harness.

AgentSkillOS operational validity. Each skill-pool condition contributes 1,440 scheduled attempts on the balanced three-model panel. The primary cross-benchmark analysis uses `skill_200`; `skill_900` is a stress condition that changes the visible skill-selection burden while keeping tasks, models, and harnesses matched. Table 26 reports the complete process summary for both conditions.

Table 27 shows that the stress condition changes valid completion differently across harnesses. Increasing the visible skill pool from 200 to 900 reduces OpenCode's model-averaged VCR by 26.7 percentage points, but increases Pi CLI's by 6.4 points. Their VCR win rates move in the same opposing directions. At the pair level, Codex CLI operates less cleanly than the additive baseline predicts for MiniMax, but more cleanly for Qwen, under both conditions. The effect of skill-pool size depends jointly on the model and the harness used for skill selection and execution.

F.7 COMPLETE RESOURCE RESULTS

Primary Token Loads and Resource Tails. Panel (b) of Figure 9 shows that Token Load is strongly right-skewed: mean load exceeds median load in 89 of 90 fully covered primary cells, and mean loads in the highlighted cells are roughly seven to twelve times their medians. Table 29 reports mean Token Load for primary configurations with complete token accounting, including the fixed system references.

Table 25: AgentSkillOS mean Task Score over four runs: 200 skills (primary), 900 (matched stress). Bold/underlined: best/second-best per core model–condition block; fixed reference excluded.

Model	Harness	200 skills (primary)	900 skills (stress)
MiniMax M3	DeepAgents	80.07	<u>83.02</u>
	OpenCode	**93.49**	64.08
	Pi CLI	<u>85.39</u>	**89.98**
	Codex CLI	36.44	36.42
Kimi K2.7	DeepAgents	<u>89.92</u>	<u>84.34</u>
	OpenCode	84.94	62.87
	Pi CLI	**95.91**	**90.87**
	Codex CLI	55.66	66.57
Qwen3.6-35B-A3B	DeepAgents	<u>74.01</u>	61.77
	OpenCode	**76.12**	50.15
	Pi CLI	56.23	**75.98**
	Codex CLI	63.06	<u>71.42</u>
GPT-5.5 xhigh + Codex CLI		97.57	89.68

Table 26: AgentSkillOS process metrics (%). A/B: VCR and model/harness summaries (pairwise win rates, means, type-7 25th percentiles, and minima). C: normal-exit and valid-output rates. Native references remain separate from balanced-panel summaries.

Panel A: `skill_200` (primary)

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI	Mean	25th pct.	Minimum	Model-side win rate
MiniMax M3	80.83	96.67	90.00	36.67	76.04	69.79	36.67	37.50
Kimi K2.7	92.50	83.33	99.17	63.33	84.58	78.33	63.33	75.00
Qwen3.6-35B-A3B	82.50	81.67	59.17	71.67	73.75	68.54	59.17	37.50
Harness mean	85.28	87.22	82.78	57.22	–	–	–	–
Harness 25th pct.	81.67	82.50	74.58	50.00	–	–	–	–
Harness minimum	80.83	81.67	59.17	36.67	–	–	–	–
VCR win rate	66.67	66.67	55.56	11.11	–	–	–	–

Panel B: `skill_900` (stress)

Model	DeepAgents	OpenCode	Pi CLI	Codex CLI	Mean	25th pct.	Minimum	Model-side win rate
MiniMax M3	86.67	64.17	92.50	41.67	71.25	58.54	41.67	50.00
Kimi K2.7	85.83	65.83	94.17	70.83	79.17	69.58	65.83	75.00
Qwen3.6-35B-A3B	71.67	51.67	80.83	79.17	70.83	66.67	51.67	25.00
Harness mean	81.39	60.56	89.17	63.89	–	–	–	–
Harness 25th pct.	78.75	57.92	86.67	56.25	–	–	–	–
Harness minimum	71.67	51.67	80.83	41.67	–	–	–	–
VCR win rate	55.56	11.11	100.00	33.33	–	–	–	–

Panel C: harness-level termination and handoff components

Condition	Harness	Normal exit	Valid output	VCR
`skill_200`	DeepAgents	93.06	85.28	85.28
	OpenCode	95.83	89.17	87.22
	Pi CLI	95.83	83.89	82.78
	Codex CLI	89.17	57.22	57.22
`skill_900`	DeepAgents	95.56	81.39	81.39
	OpenCode	91.11	63.33	60.56
	Pi CLI	98.61	89.44	89.17
	Codex CLI	93.89	64.44	63.89
`skill_200`	GPT-5.5–Codex	100.00	100.00	100.00
`skill_900`	GPT-5.5–Codex	100.00	96.67	96.67

Table 27: AgentSkillOS additive VCR interaction residual (percentage points). Residuals are centered separately within each skill-pool condition.

Condition	Model	DeepAgents	OpenCode	Pi CLI	Codex CLI
skill_200	MiniMax M3	-2.36	+11.53	+9.31	-18.47
	Kimi K2.7	+0.76	-10.35	+9.93	-0.35
	Qwen3.6-35B-A3B	+1.60	-1.18	-19.24	+18.82
skill_900	MiniMax M3	+7.78	+6.11	+5.83	-19.72
	Kimi K2.7	-0.97	-0.14	-0.42	+1.53
	Qwen3.6-35B-A3B	-6.81	-5.97	-5.42	+18.19

Table 28: Matched resource comparisons (18 complete primary model–benchmark blocks; 72 cells). First three columns: %; brackets: pointwise 95% bootstrap intervals; bold/underlined: best/second-best. Definitions and resampling: Section 4.

Harness	Token-use win rate	Outcome-preserving win rate	Pareto-frontier share	Relative load
DeepAgents	<u>51.85</u> [44.44, 53.70]	14.81 [11.11, 22.22]	50.00 [27.78, 55.56]	1.095 [1.043, 1.159]
OpenCode	44.44 [40.74, 51.85]	**35.19** [31.48, 44.44]	**61.11** [50.00, 72.22]	1.150 [1.112, 1.202]
Pi CLI	**55.56** [50.00, 61.11]	<u>24.07</u> [20.37, 33.33]	50.00 [38.89, 66.67]	<u>0.906</u> [0.827, 0.967]
Codex CLI	48.15 [44.44, 51.85]	<u>24.07</u> [16.67, 27.78]	<u>55.56</u> [44.44, 61.11]	**0.877** [0.838, 0.927]

Table 29: Mean primary Token Load (million tokens per scheduled attempt) for configurations with complete token accounting. Dashes denote configurations outside the evaluated panel. AgentSkillOS uses skill_200. Fixed references are reported as complete systems, separately from cross-harness indices.

Model	Harness	SWE-bench-45	Terminal-Bench 2	VitaBench	LOCA-Bench	ClawMark	AgentSkill OS
DeepSeek V4 Flash Preview	DeepAgents	2.358	2.484	0.933	6.181	–	–
	OpenCode	1.531	0.677	0.503	2.390	–	–
	Pi CLI	2.434	1.666	0.296	9.073	–	–
	Codex CLI	3.762	1.244	0.656	1.288	–	–
MiniMax M3	DeepAgents	6.444	1.860	0.562	6.332	6.333	2.950
	OpenCode	10.722	1.882	0.404	14.644	3.563	3.679
	Pi CLI	10.288	2.626	0.331	3.546	4.327	3.333
	Codex CLI	3.850	2.826	0.724	3.105	7.709	38.040
Kimi K2.7	DeepAgents	8.955	2.869	0.380	1.707	9.745	0.977
	OpenCode	4.451	1.471	0.301	3.737	3.217	1.630
	Pi CLI	4.849	0.834	0.229	3.280	0.056	1.567
	Codex CLI	0.446	1.692	0.587	1.514	0.173	0.467
Qwen3.6-35B-A3B	DeepAgents	2.153	0.617	0.775	1.457	4.725	1.058
	OpenCode	3.597	0.714	0.430	2.646	3.304	3.926
	Pi CLI	19.018	1.316	0.277	3.537	3.197	3.274
	Codex CLI	4.770	1.183	1.620	1.480	4.327	0.775
GPT-5.5 xhigh + Codex CLI		2.197	0.927	0.461	2.193	6.510	1.285
DeepSeek V4 Pro 0813 + DeepSeek Harness		3.117	1.515	0.030	5.080	–	–

Resource use is configuration dependent and is interpreted jointly with task outcomes and trajectory behavior. Within the same model and benchmark, harnesses can differ by more than an order of magnitude in Token Load; the lightest harness also changes across models and benchmarks. Low Token Load accompanies efficient interaction or early stopping, while high Token Load accompanies productive exploration or repeated failed actions. Table 30 reports Token Load, tokens per success, and coverage for all 111 primary, stress-condition, and fixed-reference cells.

Failure tails and fixed systems. Token use is concentrated in the failure tail. The highest-token 5% of Qwen–Pi SWE runs consume about 83% of that configuration's tokens. Failed attempts account for almost all token use; conversely, some low-token ClawMark configurations terminate early and score poorly. The fixed systems are also strongly benchmark-dependent. GPT-5.5–Codex uses about fourteen times as many tokens per attempt on ClawMark as on VitaBench. DeepSeek V4 Pro 0813–DeepSeek Harness uses over two orders of magnitude more on LOCA-Bench than on VitaBench, with complete token coverage in both cells. On Terminal-Bench 2, failed attempts consume over three-quarters of its tokens. Native model–harness pairings also have task-dependent interaction budgets. Resource controls should adapt to the model and task: long-context settings require compaction, while long-running software tasks require stagnation and repeated-action detection.

Complete configuration matrix. Table 30 gives the complete resource matrix used in Section 6.1. It contains 111 benchmark–condition–model–harness cells, including the AgentSkillOS stress condition and all fixed references. Outcome and exact token-coverage values are available for every cell. For LOCA-Bench, task-local provider totals cover all 4,800 scheduled attempts; tokens consumed before abnormal termination are retained.

Table 30: Resource results by benchmark, condition, model, and harness. Outcome: official Mean-of-4 or Task Score; Token Load: million tokens/attempt; Tokens/Success: million tokens. Coverage is the percentage of scheduled attempts with token accounting.

Model	Harness	Outcome	Token load	Tokens/ success	Coverage
SWE-bench-45					
DeepSeek V4 Flash Preview	DeepAgents	31.11	2.358	7.579	100.00
DeepSeek V4 Flash Preview	OpenCode	32.22	1.531	4.750	100.00
DeepSeek V4 Flash Preview	Pi CLI	35.56	2.434	6.845	100.00
DeepSeek V4 Flash Preview	Codex CLI	36.11	3.762	10.419	100.00
MiniMax M3	DeepAgents	31.11	6.444	20.714	100.00
MiniMax M3	OpenCode	62.78	10.722	17.079	100.00
MiniMax M3	Pi CLI	52.22	10.288	19.700	100.00
MiniMax M3	Codex CLI	30.56	3.850	12.600	100.00
Kimi K2.7	DeepAgents	15.56	8.955	57.566	100.00
Kimi K2.7	OpenCode	48.89	4.451	9.103	100.00
Kimi K2.7	Pi CLI	42.22	4.849	11.485	100.00
Kimi K2.7	Codex CLI	10.56	0.446	4.225	100.00
Qwen3.6-35B-A3B	DeepAgents	15.00	2.153	14.352	100.00
Qwen3.6-35B-A3B	OpenCode	22.22	3.597	16.184	100.00
Qwen3.6-35B-A3B	Pi CLI	25.00	19.018	76.071	100.00
Qwen3.6-35B-A3B	Codex CLI	26.67	4.770	17.889	100.00
GPT-5.5 xhigh + Codex CLI		65.56	2.197	3.351	100.00
DeepSeek V4 Pro 0813 + DeepSeek Harness		70.56	3.117	4.418	100.00
Terminal-Bench 2					
DeepSeek V4 Flash Preview	DeepAgents	45.79	2.484	5.425	100.00
DeepSeek V4 Flash Preview	OpenCode	40.45	0.677	1.674	100.00
DeepSeek V4 Flash Preview	Pi CLI	44.66	1.666	3.731	100.00
DeepSeek V4 Flash Preview	Codex CLI	37.64	1.244	3.306	100.00
MiniMax M3	DeepAgents	21.63	1.860	8.598	100.00
MiniMax M3	OpenCode	39.61	1.882	4.752	100.00
MiniMax M3	Pi CLI	33.15	2.626	7.924	100.00
MiniMax M3	Codex CLI	34.55	2.826	8.179	100.00
Kimi K2.7	DeepAgents	30.62	2.869	9.369	100.00
Kimi K2.7	OpenCode	39.04	1.471	3.767	100.00
Kimi K2.7	Pi CLI	43.82	0.834	1.904	100.00
Kimi K2.7	Codex CLI	31.74	1.692	5.331	100.00
Qwen3.6-35B-A3B	DeepAgents	26.40	0.617	2.337	100.00
Qwen3.6-35B-A3B	OpenCode	26.12	0.714	2.732	100.00
Qwen3.6-35B-A3B	Pi CLI	28.65	1.316	4.594	100.00
Qwen3.6-35B-A3B	Codex CLI	28.65	1.183	4.130	100.00
GPT-5.5 xhigh + Codex CLI		60.39	0.927	1.535	100.00
DeepSeek V4 Pro 0813 + DeepSeek Harness		46.91	1.515	3.229	100.00

Continued on next page

Table 30 continued

Model	Harness	Outcome	Token load	Tokens/ success	Coverage
VitaBench					
DeepSeek V4 Flash Preview	DeepAgents	38.00	0.933	2.455	100.00
DeepSeek V4 Flash Preview	OpenCode	45.38	0.503	1.109	100.00
DeepSeek V4 Flash Preview	Pi CLI	48.94	0.296	0.605	100.00
DeepSeek V4 Flash Preview	Codex CLI	49.06	0.656	1.338	100.00
MiniMax M3	DeepAgents	51.00	0.562	1.102	100.00
MiniMax M3	OpenCode	49.38	0.404	0.819	100.00
MiniMax M3	Pi CLI	49.25	0.331	0.672	100.00
MiniMax M3	Codex CLI	45.06	0.724	1.607	100.00
Kimi K2.7	DeepAgents	43.06	0.380	0.883	100.00
Kimi K2.7	OpenCode	44.38	0.301	0.678	100.00
Kimi K2.7	Pi CLI	40.25	0.229	0.569	100.00
Kimi K2.7	Codex CLI	41.44	0.587	1.416	100.00
Qwen3.6-35B-A3B	DeepAgents	27.69	0.775	2.799	100.00
Qwen3.6-35B-A3B	OpenCode	30.88	0.430	1.393	100.00
Qwen3.6-35B-A3B	Pi CLI	31.19	0.277	0.887	100.00
Qwen3.6-35B-A3B	Codex CLI	22.88	1.620	7.081	100.00
GPT-5.5 xhigh + Codex CLI		46.56	0.461	0.989	100.00
DeepSeek V4 Pro 0813	DeepSeek Harness	54.13	0.030	0.056	100.00
LOCA-Bench					
DeepSeek V4 Flash Preview	DeepAgents	20.67	6.181	29.910	100.00
DeepSeek V4 Flash Preview	OpenCode	31.00	2.390	7.709	100.00
DeepSeek V4 Flash Preview	Pi CLI	29.00	9.073	31.287	100.00
DeepSeek V4 Flash Preview	Codex CLI	50.67	1.288	2.542	100.00
MiniMax M3	DeepAgents	24.00	6.332	26.384	100.00
MiniMax M3	OpenCode	21.00	14.644	69.736	100.00
MiniMax M3	Pi CLI	12.67	3.546	27.992	100.00
MiniMax M3	Codex CLI	87.00	3.105	3.569	100.00
Kimi K2.7	DeepAgents	8.33	1.707	20.482	100.00
Kimi K2.7	OpenCode	27.67	3.737	13.506	100.00
Kimi K2.7	Pi CLI	34.33	3.280	9.553	100.00
Kimi K2.7	Codex CLI	54.00	1.514	2.803	100.00
Qwen3.6-35B-A3B	DeepAgents	8.67	1.457	16.810	100.00
Qwen3.6-35B-A3B	OpenCode	21.00	2.646	12.601	100.00
Qwen3.6-35B-A3B	Pi CLI	15.00	3.537	23.578	100.00
Qwen3.6-35B-A3B	Codex CLI	36.33	1.480	4.073	100.00
GPT-5.5 xhigh + Codex CLI		76.00	2.193	2.885	100.00
DeepSeek V4 Pro 0813 + DeepSeek Harness		73.33	5.080	6.927	100.00
ClawMark					
MiniMax M3	DeepAgents	54.78	6.333	–	100.00
MiniMax M3	OpenCode	69.62	3.563	–	100.00
MiniMax M3	Pi CLI	62.58	4.327	–	100.00
MiniMax M3	Codex CLI	58.49	7.709	–	100.00
Kimi K2.7	DeepAgents	58.65	9.745	–	100.00
Kimi K2.7	OpenCode	70.36	3.217	–	100.00
Kimi K2.7	Pi CLI	11.93	0.056	–	100.00
Kimi K2.7	Codex CLI	11.96	0.173	–	100.00
Qwen3.6-35B-A3B	DeepAgents	52.71	4.725	–	100.00
Qwen3.6-35B-A3B	OpenCode	63.62	3.304	–	100.00
Qwen3.6-35B-A3B	Pi CLI	52.61	3.197	–	100.00
Qwen3.6-35B-A3B	Codex CLI	61.21	4.327	–	100.00
GPT-5.5 xhigh + Codex CLI		74.31	6.510	–	100.00
AgentSkillOS (`skill_200`, primary)					
MiniMax M3	DeepAgents	80.07	2.950	–	100.00
MiniMax M3	OpenCode	93.49	3.679	–	100.00
MiniMax M3	Pi CLI	85.39	3.333	–	100.00
MiniMax M3	Codex CLI	36.44	38.040	–	100.00
Kimi K2.7	DeepAgents	89.92	0.977	–	100.00
Kimi K2.7	OpenCode	84.94	1.630	–	100.00
Kimi K2.7	Pi CLI	95.91	1.567	–	100.00
Kimi K2.7	Codex CLI	55.66	0.467	–	100.00
Qwen3.6-35B-A3B	DeepAgents	74.01	1.058	–	100.00
Qwen3.6-35B-A3B	OpenCode	76.12	3.926	–	100.00
Qwen3.6-35B-A3B	Pi CLI	56.23	3.274	–	100.00
Qwen3.6-35B-A3B	Codex CLI	63.06	0.775	–	100.00
GPT-5.5 xhigh + Codex CLI		97.57	1.285	–	100.00
AgentSkillOS (`skill_900`, stress)					
MiniMax M3	DeepAgents	83.02	4.054	–	100.00
MiniMax M3	OpenCode	64.08	6.475	–	100.00
MiniMax M3	Pi CLI	89.98	4.079	–	100.00
MiniMax M3	Codex CLI	36.42	9.668	–	100.00
Kimi K2.7	DeepAgents	84.34	2.128	–	100.00
Kimi K2.7	OpenCode	62.87	2.576	–	100.00
Kimi K2.7	Pi CLI	90.87	3.031	–	100.00
Kimi K2.7	Codex CLI	66.57	0.425	–	100.00
Qwen3.6-35B-A3B	DeepAgents	61.77	3.044	–	100.00

Continued on next page

Table 30 continued

Model	Harness	Outcome	Token load	Tokens/ success	Coverage
Qwen3.6-35B-A3B	OpenCode	50.15	4.123	–	100.00
Qwen3.6-35B-A3B	Pi CLI	75.98	4.536	–	100.00
Qwen3.6-35B-A3B	Codex CLI	71.42	0.574	–	100.00
GPT-5.5 xhigh + Codex CLI		89.68	0.906	–	100.00

G BENCHMARK-SPECIFIC EVALUATION METRICS

This appendix specifies how each benchmark's official evaluator output is converted into the task-level scores used in the main paper. Best-of-4 and stress-condition results are secondary analyses.

Operational validity status mapping. For SWE-bench-45, the normal-termination condition is a recognized normal stop (`Submitted`, `NoPatch`, or a successful exit-zero completion), and the valid-handoff condition requires the evaluator to accept the patch at its application stage. For Terminal-Bench 2, normal termination requires a clean return without a timeout or unhandled framework failure, and valid handoff requires the post-run verifier to produce a consumable result. For LOCA-Bench, normal termination requires `status=success`; valid handoff requires a finite final-state evaluator result, including one produced before a subsequently abnormal backend exit. For ClawMark, normal termination requires `status=ok`, return code zero, and no recorded framework failure; valid handoff requires a finite official score. For AgentSkillOS, normal termination requires the task runner to record `status=completed`; valid handoff requires a newly copied generated file whose path matches, or is nested under, at least one task-declared expected output path. Task correctness is not required. For VitaBench, normal termination requires `termination_reason=agent_stop`, and valid handoff requires a finite evaluator reward. In every benchmark, correctness affects the official task outcome but is never part of the valid-handoff condition.

Shared binary aggregation. SWE-bench-45, Terminal-Bench 2, and LOCA-Bench have binary run-level outcomes. For any one of these benchmarks, let N be its number of tasks and let $a_{i,r} \in \{0, 1\}$ denote whether task i succeeds on repetition $r \in \{1, 2, 3, 4\}$. We report

$$\text{Best-of-4} = \frac{100}{N} \sum_{i=1}^{N} \mathbb{I}\left[\max_{1 \le r \le 4} a_{i,r} = 1\right], \tag{6}$$

$$\text{Mean-of-4} = \frac{100}{4N} \sum_{i=1}^{N} \sum_{r=1}^{4} a_{i,r}. \tag{7}$$

Best-of-4 is the fraction of tasks solved at least once in four attempts, whereas Mean-of-4 is the success probability of a uniformly selected run.

SWE-bench-45. Each run starts from the task's prescribed repository state and is evaluated with the official task-specific regression tests. We set $a_{i,r} = 1$ only when the submitted patch fixes the designated failing tests without breaking the tests that previously passed. The shared binary aggregation above is applied with $N = 45$.

Terminal-Bench 2. The official task-specific verifier evaluates the resulting environment state, and a run succeeds only when it receives full credit. We apply the shared binary aggregation with $N = 89$.

LOCA-Bench. The final-state evaluator returns one when the resulting application state matches the task's exact target state and zero otherwise. We apply the shared binary aggregation with $N = 75$. A correct evaluator result remains an official task success even if the backend later records an abnormal exit; such a run is not a clean run because it fails the normal-termination gate.

VitaBench. VitaBench measures complete task success rather than partial progress. Its sliding-window trajectory evaluator deduplicates the expected-state and overall rubrics and carries their

satisfaction states across overlapping 10-turn windows with a two-turn overlap. For trajectory i, let $\mathcal{Q}_i$ be the resulting rubric set. Its binary reward is $R_i = \mathbb{1}[\bigwedge_{q \in \mathcal{Q}_i} q]$: the run receives one only if every rubric is satisfied.

For each environment d, we evaluate $N_d = 100$ tasks over four trials. Let $a_{d,j,r} \in \{0, 1\}$ denote the binary reward for task j in trial r. We report

$$\text{Best-of-4}_d = \frac{100}{N_d} \sum_{j=1}^{N_d} \mathbb{I}\left[\max_{1 \le r \le 4} a_{d,j,r} = 1\right], \tag{8}$$

$$\text{Mean-of-4}_d = \frac{100}{4N_d} \sum_{j=1}^{N_d} \sum_{r=1}^{4} a_{d,j,r}. \tag{9}$$

For either metric M, its Total is the unweighted macro-average over Delivery, In-store, OTA, and Cross-domain, $M_{\text{Total}} = \frac{1}{4} \sum_{d \in \mathcal{D}} M_d$.

ClawMark. Each task has a weighted rubric implemented by deterministic Python functions that inspect the resulting state of the sandboxed services. The task score is the weight-normalized credit across rubric items and lies in $[0, 1]$. For each task, Best-of-4 is the maximum continuous score across four runs and Mean-of-4 is their average. We then average both task-level quantities over all 100 tasks and report percentages. Unlike the binary benchmarks, ClawMark's Best-of-4 retains partial-credit differences below full success.

AgentSkillOS. Task Score is the official AgentSkillOS evaluator score on a 0–100 scale. We average it over four repetitions of the same 30 tasks. The cross-benchmark primary outcome uses the `skill_200` condition; the 900-skill condition is reported as a stress test. Execution status is used only for operational validity, not Task Score.

H Model–Harness Formalization

This appendix expands the runtime interfaces summarized in Section 3.

We separate the model, the harness, and the external environment. Let m and h index models and harnesses, respectively. At interaction step t, the environment has state e_t, exposes a raw signal $x_t \sim \Omega(\cdot \mid e_t)$, accepts an executable action a_t, and evolves as $e_{t+1} \sim T(\cdot \mid e_t, a_t)$, where Ω is the observation kernel and T is the transition kernel. This partially observed interaction model (Kaelbling et al., 1998) specifies the interfaces among the model, harness, and environment.

A model M_m is a conditional generator. Given model-facing context c_t, it samples an output proposal

$$y_t \sim M_m(\cdot \mid c_t),$$

The proposal y_t may contain text, structured tool-use proposals, or control suggestions, but it is not itself an executed environment action. Execution requires a runtime that constructs the context, interprets and validates the proposal, and closes the feedback loop (Yao et al., 2023; Yang et al., 2024).

We define an *agent harness* H_h as this stateful runtime layer:

$$H_h = \langle \text{Observe}_h, \text{Context}_h, \text{Control}_h, \text{Act}_h, \text{Update}_h, \text{Govern}_h \rangle, \qquad \mathsf{Agent}_{m,h} = H_h[M_m].$$

Observe_h converts raw environment feedback into model-usable observations; Context_h selects and formats instructions, history, tool descriptions, memory, and artifacts; Control_h coordinates generation, retries, recovery, delegation, and stopping; Act_h maps model proposals to tool or environment operations; Update_h persists runtime memory and artifacts; and Govern_h enforces permissions, budgets, validation, verification, and handoff. This decomposition follows the current view of a harness as a runtime substrate rather than a prompt wrapper (Guo et al., 2026; de Macedo, 2026).

The harness maintains *runtime memory* μ_t, comprising retained summaries, memory entries, artifact references, and execution bookkeeping used across interactions. This implementation-level record

retains selected interaction information and is distinct from the external environment state e_t. For runtime memory μ_t and task goal g, the observation, context, and generation interfaces are

$$o_t = \mathrm{Observe}_h(x_t, \mu_t), \qquad c_t = \mathrm{Context}_h(g, o_{\le t}, \mu_t), \qquad y_t \sim M_m(\cdot \mid c_t).$$

Here $o_{\le t}$ denotes the observation history through step t. The harness interprets and governs y_t, executes approved actions, incorporates environment feedback into memory updates $\mu_t \to \mu_{t+1}$, and decides whether to continue, recover, delegate, request input, or stop. A model invocation may lead to zero or more executed actions, whereas the environment transition equation describes a single action a_t. Repeated and parallel generations, subagents, persistent memory, and adaptive runtime memory are all instances of this same composition. The external environment and task success criteria remain outside H_h. Figure 2 illustrates these system boundaries and interfaces.