Stop Comparing LLM Agents Without Disclosing the Harness

Yunbei Zhang[*] Tulane University
Janet Wang Tulane University
Yingqiang Ge Rutgers University
Weijie Xu Independent Researcher
Jihun Hamm Tulane University
Chandan K. Reddy Virginia Tech

Project Page: yunbeizhang.github.io/harness-binding-constraint
Code: yunbeizhang/harness-binding-constraint

Abstract

This position paper argues that, for long-horizon tasks evaluated across models with comparable frontier capability, the agent execution harness, namely the infrastructure layer that governs context construction, tool interaction, orchestration, and verification around a language model, is often a stronger determinant of agent performance than the model it wraps. We formalize and defend the **Binding Constraint Thesis**: in this regime, performance variance is governed more by harness configuration than by model choice, and current evaluation protocols therefore systematically misattribute harness-level gains to model improvements. We support this thesis along three lines. First, a control-theoretic formalization treats the harness as the controller of a closed-loop dynamical system and the LLM as the stochastic policy it governs, which explains why small harness changes can produce performance shifts that exceed those obtained by substituting one model for another. Second, published benchmarks, industry deployments, and a controlled variance decomposition show that harness-induced variance can substantially exceed model-induced variance, including cases of model ranking reversal. Third, we propose a harness-aware evaluation framework with a disclosure standard and a variance decomposition protocol. Until harness specifications are disclosed, leaderboard comparisons for long-horizon agents should be treated as incomplete and potentially misleading.

1 Introduction

The standard practice in LLM agent evaluation reports a single number per $\{\texttt{model}, \texttt{benchmark}\}$ pair and attributes it to the model. Leaderboards such as SWE-bench [13], Terminal-Bench [21], AgentBench [18], and GAIA [22] treat agent performance as if it were a property of the model alone, and this convention directs research toward model scaling, fine-tuning for tool use, and prompt-level techniques. The implicit assumption is that agent capability is primarily driven by model capability, and that a sufficiently capable model produces reliable behavior on long-horizon tasks.

This assumption omits the *execution harness*: the software layer between the model and the task that constructs the context the model sees, mediates its tool calls, validates its outputs, and decides when to retry, escalate, or stop. Every benchmark score is jointly produced by a model and a harness, but the harness is rarely disclosed and almost never held constant across comparisons. The phenomenon is not anecdotal: holding the model fixed while changing only the harness raises Terminal-Bench 2 pass@1 from 69.7% to 77.0% [17], and independent third-party benchmark monitoring reports up to

[*]Corresponding author: yzhang111@tulane.edu.

Accepted to the ICML 2026 Agents in the Wild (AIWILD) Workshop.

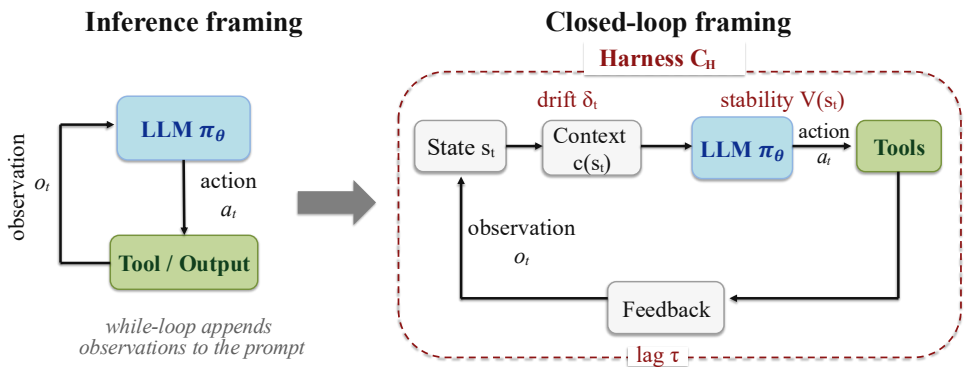

Figure 1: **From inference to control.** The inference framing (left) treats the agent as a model in a while-loop, attributing performance to π_θ. The closed-loop framing (right) makes the harness the controller $\mathcal{C}_H$: stability $V(s_t)$, context drift δ_t, and control lag τ are controller properties, the model is open-loop, and the harness closes the loop.

15 percentage points of scaffold-only variation on SWE-bench Verified [4]. On such long-horizon tasks, the same model under a different harness can therefore rank above or below a competitor.

We argue that benchmark scores for LLM agents on long-horizon tasks are not valid for cross-model comparison unless the execution harness is disclosed. Valid comparisons require either a locked-harness protocol, where a single specified harness is applied to all models, or a factorial protocol, where harness choice is varied as a controlled factor and variance components are reported. This disclosure extends to the infrastructure layer responsible for context construction, tool-call mediation, and output verification. Without such disclosure, leaderboards measure a mixture of model and harness, and the conclusions drawn from them are systematically misattributed to the model.

The practical consequences are concrete. Cross-paper model comparisons are unreliable, since two reported scores typically reflect different harnesses. Reproductions of a published score are difficult without a full harness specification, because the harness can be as important to the outcome as the model itself. Research incentives are also distorted: investments target model-side improvements while harness changes, the larger source of performance variance on long-horizon tasks, receive proportionally less attention. Better models still matter, and harness engineering is part of the deployed agent. The position calls for an attribution framework that decides whether an observed benchmark gain should be credited to the model, the harness, or their interaction. Beyond academic accuracy, these misattributions shape decisions the field is now making at scale, including which models to fund, which research directions to scale, and which agents to deploy. An evaluation regime that cannot separate model from harness routes those decisions to the wrong lever.

2 Why Current Evaluation Conflates Model and Harness

2.1 How the Harness Enters Every Benchmark Score

Standard agent evaluation queries the model with a prompt assembled from the task description, system prompt, tool catalogue, and prior trajectory state [43, 41, 39], parses the token-level output as a final answer, tool invocation, or malformed string, then routes tool calls, handles errors, compresses context as the trajectory grows, and stops after some number of steps. Every step in this pipeline (context construction, tool invocation, parsing, retry, summarization, and stopping) is a harness decision [17, 15, 19, 45]. A single benchmark score is therefore the joint outcome of these choices and the model that generates outputs against them, but the published number records the model and elides the rest.

The conflation is not hypothetical. Under the standardized SEAL scaffold on SWE-bench Pro, Claude Opus 4.5 reaches 45.9%, while under Claude Code the same model reaches 55.4%. Adding a single search subagent (WarpGrep) to otherwise identical infrastructure flips the SWE-bench Pro ordering between MiniMax 2.5 and Claude Opus 4.6, despite Claude Opus ranking higher on most other benchmarks [25]. The Holistic Agent Leaderboard (HAL) reports double-digit gaps for the same

model under different scaffolds on SWE-bench Verified Mini, with reported single-model swings of up to nearly 48 percentage points across leading frontier models [14]. Independent third-party benchmark monitoring reports up to 11 percentage points of scaffold-only variation for GPT-5 and 15 points for Kimi K2 Thinking on SWE-bench Verified [4]. None of these effects are model upgrades. They are harness substitutions, and they routinely dwarf the 2 to 4 percentage point shifts that papers report as meaningful model advances. The resulting attribution gap matters because the same composite number is being read in three incompatible ways: as a model property in cross-paper model comparisons, as a benchmark property in tracking field progress, and as an agent property in deployment decisions. However, only the third reading is well-defined. The first two require disentanglement that current reporting does not provide.

2.2 The Structural Reason: Agents Are Closed-Loop Systems

The conflation reflects a structural fact about how agents work. Agent execution is a discrete-time closed-loop dynamical process. Let s_t denote the agent's state at step t, comprising the full context window and any persistent memory. The model acts as a stochastic policy $a_t \sim \pi_\theta(\cdot \mid c(s_t))$, where c is the context construction function implemented by the harness. The harness implements a controller $\mathcal{C}$ that updates state from (s_t, a_t, o_t) to s_{t+1}, where o_t is the environment observation, filtered and validated before being integrated into state. Different harness configurations H induce different controllers $\mathcal{C}_H$ over the same model π_θ. In this model, the LLM is open-loop. It has no direct access to s_t, only to $c(s_t)$, the projection of state into context that the harness chooses to expose. It retains no memory across steps beyond what $\mathcal{C}_H$ injects into $c(s_{t+1})$. It has no mechanism for self-correction beyond the feedback path that $\mathcal{C}_H$ constructs from o_t. Adaptation, error recovery, and long-horizon coherence are properties of $\mathcal{C}_H$, not π_θ. This view continues a line of agent research that treats reasoning, action, and interfaces as coupled [43, 16, 28, 5, 40].

This asymmetry is the structural reason for the conflation. In closed-loop systems, the controller, not the open-loop policy, governs three quantities that determine long-horizon reliability. **Stability** is whether expected progress toward a goal is non-decreasing under the controller's update rule. **Context drift** is how fast task-relevant information leaves the context window. **Control lag** is the number of steps between an anomaly being detected and a corrective signal reaching the policy. None of the three is a model property. A more capable model can lower the base rate at which anomalies arise, but the response to anomalies once they occur is a property of $\mathcal{C}_H$ that does not improve when the model is upgraded [34].

The lesson is broader than agents. Open-loop systems do not become reliable through bigger actuators. They become reliable through feedback control. The same structural fact applies to LLM agents on long-horizon tasks. Fig. 1 illustrates the contrast: under the inference framing the model carries the reliability burden alone, while under the closed-loop framing the harness controller surrounds the policy and supplies the feedback that long-horizon execution requires.

2.3 Why Standard Responses Fall Short

If the conflation is structural, the question is what to do about it. The field has converged on four classes of response. Each addresses part of the problem, but none resolves the attribution question.

Standardization on a single harness. Locking the harness across submissions is the most direct response, instantiated by HAL across nine benchmarks [14], by the SWE-bench Pro unified scaffold [8, 25], and by Epoch AI's mini-SWE-agent recommendation [4]. We endorse locked-harness protocols as one of two valid evaluation regimes (Sec. 3 names the other), but any standardized harness embeds design choices that interact with model properties and forecloses the harness-side gains current evidence shows are large. The position calls for disclosure of the harness in use, not uniformity across all evaluation.

Scaling the model. A second response holds that more capable models will manage their own controllers. The empirical trend goes the other way: Claude Code redesigned rather than removed harness components as capability grew [33], the managed-agents architecture explicitly decouples model and harness because harness assumptions go stale faster than models improve [20], and AHE produces double-digit pass-rate gains on top of frontier base models [17]. No model upgrade compensates for the absence of feedback structure in the controller (Sec. 2.2).

Adding more benchmarks. A third response is that with enough independent benchmarks, harness-induced noise will average out. Recent additions span freelance software work, ML engineering, contamination-resistant coding, code synthesis, visual front-end issues, and professional services [23, 6, 12, 46, 42, 38]. The bet fails: each benchmark uses its own undisclosed harness, and harness contributions do not have zero mean, so adding benchmarks adds independent $\{model, harness\}$ draws without resolving attribution within any of them. APEX-Agents finds the opposite of what the bet requires: agent failures are predominantly execution and orchestration problems rather than knowledge failures [38].

Optimizing the harness. A fourth response is that the field already addresses harness quality directly. ADAS searches over agentic systems [11], Meta-Harness and AutoHarness optimize harness code via outer-loop search [15, 19], and AHE evolves coding-agent harnesses with cross-family transfer [17]. Agentic Context Engineering optimizes context construction as a separate surface [45], and practitioner deployment reports reach the same conclusion [30, 47]. These contributions document the phenomenon this paper analyzes but answer a different question: how to make a harness better, not how to attribute observed gains. Some report results under their own harness, compounding rather than resolving attribution. Harness optimization is part of the evidence base for the position, not a substitute for it.

The structural problem requires a structural solution: disclosure. The harness used to produce a benchmark score must be part of the experimental condition, and cross-model comparisons must hold the harness fixed (locked-harness protocol) or vary it as a controlled factor (factorial protocol).

3 The Binding Constraint Thesis

The Binding Constraint Thesis

For LLM agents operating on long-horizon tasks with comparable frontier models, let $B(M, H)$ denote the benchmark score of model $M \in \mathcal{M}$ under harness $H \in \mathcal{H}$. Define:

$$\mathrm{HV}(M) = \mathrm{Var}_{H \sim P(\mathcal{H})}[B(M, H)] \qquad \mathrm{MV}(H) = \mathrm{Var}_{M \sim P(\mathcal{M})}[B(M, H)]$$

The Binding Constraint Thesis asserts that, in this regime, HV is often comparable to or larger than MV, and may dominate it in many current long-horizon agent evaluations. Current benchmark protocols report $B(M, H^*)$ for a single undisclosed H^*, rendering HV unmeasurable and model comparisons incomplete and potentially misleading.

3.1 Variance Decomposition

Total performance variance over a population of $\{M, H\}$ draws decomposes exactly as

$$\mathrm{Var}\big(B(M,H)\big) = \underbrace{\mathrm{Var}_M\big[\mathbb{E}_H\big[B(M,H)\big]\big]}_{\mathrm{MV}} + \underbrace{\mathrm{Var}_H\big[\mathbb{E}_M\big[B(M,H)\big]\big]}_{\mathrm{HV}} + \underbrace{\mathrm{Var}\big(\text{model} \times \text{harness}\big)}_{\text{interaction}}. \quad (1)$$

The thesis asserts that the second and third terms together dominate the first in the regime considered. The interaction term is non-negligible: under the closed-loop account, a harness emphasizing self-verification helps a model with high false-confidence rates more than a model with conservative output distributions, so the same harness shift produces different gains across models. Treating model rankings as model properties requires the interaction term to be small relative to MV, a condition the thesis denies. The decomposition is the methodological core of the position. A *locked-harness* protocol fixes $H = H^*$ and recovers a clean ranking of $\{B(M_i, H^*)\}$ across models, but only under that specific H^*. A *factorial protocol* designs a grid varying both M and H, reports $\overline{\mathrm{HV}} = \mathbb{E}_M[\mathrm{HV}(M)]$ and $\overline{\mathrm{MV}} = \mathbb{E}_H[\mathrm{MV}(H)]$ averaged across the opposite axis, the interaction term, and the count of model-pair ranking reversals across harnesses, and supports interaction analysis directly.

3.2 Reliability Quantities of the Harness Controller

Stability, context drift, and control lag are the three structural channels through which harness configuration produces variance. Each is a property of $\mathcal{C}_H$, not of π_θ, and each maps to an empirical failure mode that current evaluation captures only as a single composite score. Together they identify what HV is variance of: a harness shift that changes any of the three shifts $B(M, H)$ for fixed M.

Definition 1 (Stability)**.** *An agent run is stable if the Lyapunov-like measure $V(s_t) = d(s_t, \Omega^*)$ is nonincreasing in expectation under $\mathcal{C}$, where Ω^* is the set of goal-consistent states and d is an appropriate task-specific distance measure. Formally: $\mathbb{E}[V(s_{t+1}) \mid s_t] \leq V(s_t)$ for all t. Instability manifests empirically as hallucination spirals, context overflow, and execution runaway, all of which are controller failures rather than policy failures.*

Definition 2 (Context Drift)**.** *Let p_t denote the distribution of context-window embeddings at step t, obtained by encoding the full context window using a fixed embedding model. Context drift is $\delta_t = D_{KL}(p_t \| p_0)$, the divergence of the current context distribution from the initial task-relevant state p_0. A high drift rate $\dot{\delta} = \delta_t - \delta_{t-1}$ indicates that the controller is failing to maintain task-relevant information in the context window and empirically correlates with task abandonment and semantic inconsistency over long horizons. Drift is a property of the harness configuration H through its context construction policy c. No property of the model can prevent context drift when the controller fails to manage it.*

Definition 3 (Control Lag)**.** *We propose control lag τ as a harness-specific reliability measure: the number of steps between harness anomaly detection (a tool failure, malformed output, or empty result) and the arrival of corrective action at the policy through the feedback path. Formally, if an anomaly is detected at step t_d and the corrective observation o_{t_c} reaches the policy at step t_c, then $\tau = t_c - t_d$. High τ enables failure cascades: the policy continues to act on a corrupted state for τ steps while the harness processes the error, compounding the damage before correction arrives.*

These three quantities sit on the structural side of the variance decomposition. Two configurations H_1, H_2 that differ in self-verification (affecting stability), context compression (affecting drift), or anomaly recovery (affecting control lag) each shift $B(M, H)$ for fixed M, contributing independently to HV. A more capable model can lower the base rate at which anomalies arise, which reduces demand on $\mathcal{C}_H$, but the response to anomalies once they occur is a property of $\mathcal{C}_H$ that does not improve when the model is upgraded. This is the structural reason the thesis is plausible. We use these quantities as conceptual tools for reasoning about $\mathcal{C}_H$, and develop their operational counterparts as trajectory-level metrics below.

3.3 Scope, Falsifiability, and Implications

Scope. The thesis is restricted to long-horizon tasks and comparable frontier models. Long-horizon means tasks where multiple steps of tool use, error recovery, and context management are required to reach a final outcome. The closed-loop reliability quantities only have an opportunity to act over such horizons. Comparable frontier models means a regime in which model capability gaps are not so large that MV mechanically dominates regardless of harness configuration. What can be said precisely is what falls outside the regime: short-horizon tasks and configurations where one model is dramatically more capable than the others on the relevant skills. The regime that remains is the regime where benchmark leaderboards drive research and product decisions.

Falsifiability. The thesis is empirical. It can be falsified by a factorial experiment that varies $\mathcal{H}$ while holding $\mathcal{M}$ fixed, separately varies $\mathcal{M}$ while holding $\mathcal{H}$ fixed, and finds $\overline{\text{MV}} > \overline{\text{HV}}$ on a long-horizon task distribution at comparable model capability. The thesis is not vacuous: such an outcome is conceivable in principle, and our controlled grid makes the variance ordering an empirical question. Concurrent psychometric work that decomposes agent ability into independent LLM and scaffold components offers an external probe in the same direction [9]. The thesis would be in tension with that line of work if scaffold-ability variance were systematically smaller than LLM-ability variance across benchmarks, and consistent with it otherwise.

4 Evidence

We support the thesis with two kinds of evidence: observational data from public leaderboards and a controlled factorial experiment under matched conditions.

Table 1: Harness-induced performance changes on agent benchmarks (model held fixed, only infrastructure changes).

Benchmark	Model (Fixed)	Harness Change	Layer	Δ
SWE-bench Pro	Claude Opus 4.5	SEAL → Claude Code (45.9% → 55.4%)	E, T, C	+9.5pp
SWE-bench Verified	Grok 4	SWE-agent → xAI scaffold (58.6% → 72–75%)	T, C, O	+14–16pp
TerminalBench 2.0	Fixed model	Prompt + middleware + verif. (52.8% → 66.5%)	C, S, V	+13.7pp
TerminalBench-2	Fixed model	Automated harness opt.	All	76.4%
TerminalBench 2	GPT-5.4 (high)	AHE harness evolution (69.7% → 77.0%)	C, T, S, O	+7.3pp
Coding benchmarks	Fixed model	Context format + tool defs	T, C	∼10×
Agent tasks (Vercel)	Fixed model	15 tools → 2 tools	T	80%→100%
SWE-bench Pro	Various	+WarpGrep search subagent	T	+2.1–2.2pp

Layers: E = Execution, T = Tool, C = Context, S = Scheduling, O = Observability, V = Verification.
Sources: SWE-bench Pro data from Morph [25]. SWE-bench Verified scaffold reports from Vals AI and Anthropic [37, 32]. Terminal-Bench harness results from Trivedy, Meta-Harness, and AHE [36, 15, 17]. Tool-reduction case from Vercel [29]. Edit-tool harness result from Bölük [3].

4.1 Evidence from Public Leaderboards

SWE-bench Pro under standardized vs varied scaffolds. On the Morph leaderboard [25], the six leading frontier models span only 4.9 percentage points under the standardized SEAL scaffold (41.0% to 45.9%). Holding Claude Opus 4.5 fixed and varying only the harness widens this to 9.5 points (SEAL vs Claude Code). The within-model harness range thus exceeds the within-harness model range by roughly twofold, a lower bound on the HV / MV ratio for this slice. Adding the WarpGrep search subagent on top of identical infrastructure adds 2.1 to 2.2 points across models, comparable to a routine model upgrade and sufficient to flip the MiniMax 2.5 vs Claude Opus 4.6 ordering: a single tool addition makes the interaction term visible.

Cross-scaffold gaps on SWE-bench Verified Mini. HAL reports double-digit cross-scaffold gaps under SWE-Agent versus HAL Generalist, of 34 points for Claude Sonnet 4.5 (68% to 34%), 34 points for GPT-5 Medium (46% to 12%), and nearly 48 points for o4-mini [14], much larger than the cross-model spread under either scaffold individually. The same psychometric work [9], in its leaderboard analysis, reaches the same qualitative conclusion under an item-response-theory model.

Third-party benchmark monitoring. Independent monitoring reports up to 11 to 15 percentage points of scaffold-only variation on SWE-bench Verified, with the explicit observation that scaffold choice has the single biggest impact on overall performance [4]. The recommendation to use mini-SWE-agent for cross-model comparison is one realization of the locked-harness protocol. The thesis adds that the measurement gap between locked and unlocked scaffolds is what the variance decomposition makes visible.

The pattern is consistent and large. But public evidence is observational. Public harnesses differ across many dimensions and were built by different teams with different engineering budgets, so the public data do not isolate which harness components drive the variance.

4.2 Controlled Factorial

To isolate harness-controlled variance, we vary configurations along the three reliability quantities while holding all other factors fixed, and report the resulting decomposition on a representative long-horizon coding task distribution.

Setup. We evaluate three frontier models, GPT-5.4 [27], Kimi K2.6 [24], and GLM-5.1 [44], selected because they were tightly clustered on the LLM Stats coding leaderboard,[2] across three harness

[2] LLM Stats coding leaderboard, https://llm-stats.com/, accessed April 23, 2026.

Table 2: Benchmark scores $B(M, H)$ on SWE-bench Verified subset100. Cells report mean pass@1 percentages over two runs, with percent signs omitted. Run-level results are reported in Appx. B, Table 7. The HV column and MV row report variances in pp^2.

	H_1 (Minimal)	H_2 (Improved)	H_3 (Full)	**HV**(M)
GLM-5.1	52.5	56.5	65.5	**29.56**
GPT-5.4	55.0	58.5	63.5	**12.17**
Kimi K2.6	52.0	59.0	60.5	**13.72**
MV(H)	**1.72**	**1.17**	**4.22**	

Aggregate $\overline{\text{HV}}/\overline{\text{MV}}$ ratio: 7.80×. Ranking reversal pairs: 6 out of 9 model-pair/harness-pair comparisons.

configurations $\{H_1, H_2, H_3\}$ on a difficulty-stratified 100-task subset of SWE-bench Verified [13]. Each (M_i, H_j) cell uses two independent runs with shared task order, Docker execution environment, SWE-bench evaluation pipeline, 50-step budget, and 120-second per-step timeout. Subset construction, model-selection rationale, and full harness specifications are in Appx. B.

The three harness configurations vary deliberately along the three reliability quantities:

- H_1 **(Minimal)**: no context compression, verbose tool schema, no retry logic, no verification hooks, and no anomaly recovery. High drift rate, high control lag, and no stability guarantees. This is the baseline open-loop harness.
- H_2 **(Improved)**: compressed context with task-relevance retrieval, minimal tool schema, and structured retry on tool failure with exponential backoff. This configuration reduces drift and provides moderate control lag with basic closed-loop feedback.
- H_3 **(Full)**: H_2 plus per-step self-checking, KL-style drift checks every five steps, anomaly-detection middleware for repeated action loops and context contradictions, full output validation, and checkpoint rollback retaining the last 10 states. This configuration has low drift, low control lag, and explicit stability enforcement.

Metrics. We report pass@1 $B(M_i, H_j)$, model-induced variance per harness $\text{MV}(H_j) = \text{Var}_M[B(M, H_j)]$, harness-induced variance per model $\text{HV}(M_i) = \text{Var}_H[B(M_i, H)]$, the aggregate ratio $\overline{\text{HV}}/\overline{\text{MV}}$ averaged across the opposite axis, and the count of model-pair ranking reversals across harnesses.

Findings. The grid is consistent with the Binding Constraint Thesis on this task distribution. Average HV is 18.48 pp^2 versus average MV of 2.37 pp^2, a ratio of 7.80×. Changing the harness moves GLM-5.1 by 13.0 percentage points and GPT-5.4 and Kimi K2.6 by 8.5 points each. Changing the model within a fixed harness moves scores by only 3.0, 2.5, and 5.0 points for H_1, H_2, H_3. The interaction term is visible as six ranking reversals across the nine possible model-pair / harness-pair comparisons. We do not claim that the 7.80× ratio is universal. The grid demonstrates that harness variance can dominate model variance under controlled conditions and that the decomposition protocol produces interpretable estimates on a realistic task distribution.

Trajectory-log analysis confirms that the $H_1 \rightarrow H_2$ shift reduces control noise rather than increasing model knowledge, while the $H_2 \rightarrow H_3$ gain comes primarily from closing the verification and recovery loop. The per-paragraph mechanism analysis and the cross-model variance pattern across H_1, H_2, H_3 are in Appx. E.

5 A Harness-Aware Evaluation Framework

We argue that long-horizon agent evaluation should adopt a harness-aware framework with three components: a structured disclosure card, a variance decomposition protocol, and a short set of trajectory-level metrics that together make harness variance reportable, interpretable, and attributable.

The harness card. Every agent benchmark submission should include a *Harness Card*: a structured disclosure of the harness configuration organized by the seven-layer ETCSOVG taxonomy, intended as an attention checklist rather than a design constraint. The seven layers cover **E**xecution (runtime substrate, sandboxing, step and task budgets), **T**ool (tool list, schemas, error contract), **C**ontext

(window cap, compression and retrieval policy, persistent memory), **S**cheduling (agent loop, retry and escalation rules), **O**bservability (logged artifacts and traces), **V**erification (validation, self-checking, anomaly detection), and **G**overnance (permission model, side-effect boundaries, human approval points). Field-level disclosure requirements per layer are in Appx. A (Table 3). The goal is to make configuration visible, not to standardize it, so that a reader comparing two cards can locate whether a score difference is attributable to the model, the harness, or their interaction.

Variance decomposition protocol. The variance decomposition becomes a protocol when an evaluation runs the same model under at least two meaningfully different harnesses and the same harness under at least two models on a fixed task set. The minimal valid design is a 2×2 model-by-harness grid with task order, execution environment, evaluation script, API parameters, and stopping rules held constant. A harness difference counts as meaningful only if it changes at least one ETCSOVG layer in a way expected to affect stability, drift, or control lag, for example retrieval-based context compression, a change to the tool schema and error format, or added verification and recovery hooks. For a designed $\{M_i\} \times \{H_j\}$ grid, we recommend reporting four statistics alongside the headline score: HV per model and MV per harness, the aggregate ratio $\overline{\text{HV}}/\overline{\text{MV}}$, the count of model-pair ranking reversals across harnesses, and the partial eta-squared coefficient

$$\eta_p^2 = \frac{SS_{\text{interaction}}}{SS_{\text{interaction}} + SS_{\text{error}}} \tag{2}$$

where SS denotes the sum of squares from a fixed-effects two-way ANOVA. Three caveats apply: η_p^2 is a fixed-effects quantity on the chosen grid (if harnesses are sampled from a larger population the appropriate quantity is the variance component σ^2_{MH} from a mixed-effects model), it is positively biased in small grids and should be reported alongside ω^2 or a bootstrap interval, and a large η_p^2 is neither necessary nor sufficient for ranking flips, so it should be paired with explicit reversal counts.

Trajectory-level metrics. Aggregate pass rates leave open which controller-side mechanism is producing HV when it is large. We propose three trajectory-level metrics that operationalize the three reliability quantities and let HV shifts be associated with specific harness layers. We do not estimate these metrics on the 3×3 grid. Instead, we specify them as reporting requirements for future harness-aware evaluations. *Recovery Rate* $RR(k)$ operationalizes Stability as the probability that a trajectory transitions from a detected anomaly state (tool error, malformed output, failed validation, rejected patch) back to a task-advancing state within k steps. Reporting RR as a curve over $k \in \{1, 3, 5, 10\}$ is preferable to a single number because the choice of k trades promptness against eventual recovery. *Context Retention* operationalizes Context Drift as the fraction of task-relevant files, tests, and constraints present in the constructed context at each step, an auditable proxy that is monotone with respect to δ_t. *Control Lag* τ is measured directly: the number of steps between a harness-detected anomaly and the arrival of a corrective signal at the policy. Together with the variance decomposition, these metrics support attribution rather than just measurement: a shift in RR implicates verification and recovery layers, a shift in τ implicates observability and tool layers, and a shift in Context Retention implicates the context construction policy. The resulting attribution is descriptive co-movement, not causal. Layer-ablation designs that toggle a single ETCSOVG layer at a time are the natural extension once these metrics are routinely collected. Two complementary diagnostics extend the framework as future work for the community: *Schema Compliance Rate* measures the fraction of model outputs that parse without harness intervention (interface alignment), and *Action Efficiency* measures the ratio of productive, task-advancing actions to total actions (trajectory productivity). Both supplement, rather than substitute for, the three reliability quantities.

6 Alternative Views and Counterarguments

Four counterarguments deserve a direct response: that scaling models will dissolve the problem, that standardization is the right reply, that interaction effects are small enough to ignore, and that the harness, as part of the deployed agent, cannot be cleanly separated in evaluation.

Model capability will dissolve the harness problem. One might argue that sufficiently capable models will manage their own context and recovery, removing the need for external infrastructure. The empirical trend goes the other way: the harness ecosystem has grown more complex as models have improved, with Claude Code redesigning rather than removing harness components, the managed-agents architecture explicitly decoupling model and harness, and AHE producing double-digit gains on top of frontier base models [33, 20, 17]. The closed-loop account explains why: long-horizon

reliability is a controller property, and no model upgrade compensates for the absence of feedback structure.

Standardization is enough. One might concede that harness variance is large but argue for standardizing the harness across the field rather than rethinking reporting. We partially endorse it: locked-harness protocols are one of the two valid regimes. But any standardized harness embeds design choices that interact with model properties, as Epoch AI itself flags when recommending mini-SWE-agent [4], and locking the harness forecloses harness-side gains that current evidence shows are large [17, 15, 19]. Standardization is also unlikely to be uniform in practice: agent harnesses manage stateful execution across multiple layers, and vendors have strong competitive incentives to differentiate at the runtime layer. Whether the harness is locked or varied, the position is that the configuration must be disclosed and treated as part of the experimental condition.

Interaction effects are small enough to ignore. One might concede that HV is large but argue that the interaction term is small, so any one harness still recovers a stable ordering. SWE-bench Pro data refute this: under SEAL the top six frontier models span only 4.9 points, and adding the WarpGrep subagent flips MiniMax 2.5 versus Claude Opus 4.6 [25]. The LangChain Terminal-Bench result shows the same coupling, where a harness tuned around one model yields a lower score when paired with another [36]. Our controlled grid confirms this under matched conditions: six ranking reversals across nine model-pair / harness-pair comparisons.

The harness is part of the deployed agent, so separating is artificial. One might argue that since deployed agents are (model, harness) compositions, attributing performance to one factor is a category error. This argument is correct as a description of deployment, but deployment optimizes the agent as a whole while *evaluation that compares models* must isolate the model contribution. Harness improvements are real gains that deployments rightly capture. The position is that benchmark scores read as model rankings must disclose the harness, because otherwise the rankings are not informative about the quantity readers are trying to measure. The disclosure standard and variance decomposition protocol keep deployment-style and model-comparison-style attribution from being conflated.

7 Conclusion and Discussion

The position has three audiences. For *researchers*, reviewers should ask "what harness was used?" as routinely as they ask about hyperparameters and decoding settings, since a model score without a harness specification is missing part of the experimental condition. For *benchmark designers*, agent benchmarks should expose harness variation as a first-class evaluation dimension, either through locked-harness tracks for clean model comparison or factorial tracks that report variance decompositions, building on programs underway in HAL [14], the unified scaffold of SWE-bench Pro [8], and standardization recommendations from Epoch AI [4]. For *practitioners*, model selection alone is an incomplete optimization loop: in our controlled grid, moving from H_1 to H_3 shifts pass@1 by 8.5 to 13.0 percentage points at a fixed model, while changing the model at a fixed harness shifts pass@1 by 2.5 to 5.0 points. The engineering object is not only the model endpoint but the context, tool, recovery, verification, and governance surfaces through which the model operates.

Open questions. The position raises four questions that remain open for the community. First, the operational definition of comparable frontier models is currently informal and anchored to leaderboard proximity that is itself harness-confounded, so a non-confounded test for comparability is a prerequisite for cross-paper meta-analysis of HV / MV ratios. Second, harness diversity needs a principled notion of distance between configurations, without which $\mathrm{Var}_H[B(M, H)]$ depends on the sampling distribution $P(\mathcal{H})$ in ways not yet standardized. Third, the right balance between disclosure and locked-harness evaluation is a community decision rather than a purely technical one, since disclosure alone may not address adversarial harness selection while locking the harness slows infrastructure innovation. Fourth, the trajectory-level diagnostics admit natural extensions for the community to develop: *Schema Compliance Rate* and *Action Efficiency* would supplement the three reliability quantities as interface-alignment and productivity probes, and the perturbation stress-test detailed in Appx. C would test whether controller-side mechanisms reduce output instability under matched perturbations.

The execution harness is the binding constraint on long-horizon LLM-agent performance in the regime where benchmark leaderboards drive research and product decisions. The model is an open-loop policy and the harness is the controller that closes the loop, so reliability over a horizon is a

controller property. Long-horizon agent evaluation should disclose Harness Cards, report variance decompositions when feasible, and include trajectory-level metrics that make recovery, drift, and control lag visible. Until those practices are routine, leaderboard comparisons for long-horizon agents should be treated as incomplete and potentially misleading.

References

[1] Anomaly. Opencode: The open source coding agent., 2025. URL https://github.com/anomalyco/opencode.

[2] Anthropic. Claude-code, 2025. URL https://github.com/anthropics/claude-code.

[3] Can Bölük. I improved 15 LLMs at coding in one afternoon. Only the harness changed., February 2026. URL https://blog.can.ac/2026/02/12/the-harness-problem/.

[4] Florian Brand and JSD. Why benchmarking is hard. Epoch AI, Gradient Updates, December 2025. URL https://epochai.substack.com/p/why-benchmarking-is-hard.

[5] Yuxuan Cai, Lu Chen, Qiaoling Chen, Yuyang Ding, Liwen Fan, Wenjie Fu, Yufei Gao, Honglin Guo, Pinxue Guo, Zhenhua Han, et al. Nex-n1: Agentic models trained via a unified ecosystem for large-scale environment construction. *arXiv preprint arXiv:2512.04987*, 2025.

[6] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*.

[7] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, April 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

[8] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

[9] Chris Ge, Daria Kryvosheieva, Daniel Fried, Uzay Girit, and Kaivalya Hariharan. Agent psychometrics: Task-level performance prediction in agentic coding benchmarks. *arXiv preprint arXiv:2604.00594*, 2026.

[10] Harbor. Terminus-2, 2026. URL https://www.harborframework.com/docs/agents/terminus-2.

[11] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

[12] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

[13] 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? *arXiv preprint arXiv:2310.06770*, 2023.

[14] Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, et al. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. *arXiv preprint arXiv:2510.11977*, 2025.

[15] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[16] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pages 2219–2230, 2026.

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

[18] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[19] 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. *arXiv preprint arXiv:2603.03329*, 2026.

[20] Lance Martin, Gabe Cemaj, and Michael Cohen. Scaling Managed Agents: Decoupling the brain from the hands, April 2026. URL https://www.anthropic.com/engineering/managed-agents.

[21] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

[22] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

[23] Samuel Miserendino, Michele Wang, Tejal Patwardhan, and Johannes Heidecke. Swe-lancer: Can frontier llms earn $1 million from real-world freelance software engineering? In *International Conference on Machine Learning*, pages 44412–44450. PMLR, 2025.

[24] Moonshot AI. Kimi-k2.6, April 2026. URL https://huggingface.co/moonshotai/Kimi-K2.6.

[25] Morph. SWE-Bench Pro Leaderboard (2026): Why 46% Beats 81%, March 2026. URL https://www.morphllm.com/swe-bench-pro.

[26] OpenAI. Codex cli, 2025. URL https://developers.openai.com/codex/cli.

[27] OpenAI. Introducing gpt-5.4, March 2026. URL https://openai.com/index/introducing-gpt-5-4/.

[28] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

[29] Andrew Qu. We removed 80% of our agent's tools, December 2025. URL https://vercel.com/blog/we-removed-80-percent-of-our-agents-tools.

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

[31] Nous Research. Hermes agent — the agent that grows with you, 2026. URL https://hermes-agent.nousresearch.com/.

[32] Erik Schluntz. Raising the Bar on SWE-bench Verified with Claude 3.5 Sonnet, January 2025. URL https://www.anthropic.com/research/swe-bench-sonnet.

[33] Thariq Shihipar. Seeing like an agent: How we design tools in Claude Code, April 2026. URL https://claude.com/blog/seeing-like-an-agent.

[34] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[35] Qwen Team. Qwen3.6-plus: Towards real world agents, April 2026. URL https://qwenlm.github.io/blog/qwen3.6/.

[36] Vivek Trivedy. Improving Deep Agents with harness engineering, February 2026. URL https://www.langchain.com/blog/improving-deep-agents-with-harness-engineering.

[37] Vals AI. SWE-Bench, 2026. URL https://www.vals.ai/benchmarks/swebench.

[38] Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, et al. Apex-agents. *arXiv preprint arXiv:2601.14242*, 2026.

[39] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[40] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying llm-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE): 801–824, 2025.

[41] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

[42] John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. SWE-bench multimodal: Do AI systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=riTiq3i21b.

[43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[44] Z.ai. Glm-5.1, April 2026. URL https://huggingface.co/zai-org/GLM-5.1.

[45] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eC4ygDs02R.

[46] Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, David Lo, Binyuan Hui, Niklas Muennighoff, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

[47] Gregor Zunic. The bitter lesson of agent harnesses, April 2026. URL https://browser-use.com/posts/bitter-lesson-agent-harnesses.

A Example ETCSOVG Disclosure Card

Table 3 gives the full field set that we expect a benchmark submission to disclose. The compact example in Table 4 instantiates these fields for H_3 in our controlled experiment.

Table 3: ETCSOVG disclosure fields. A submission can provide more detail, but omitting any of these fields makes it difficult to separate model effects from harness effects.

Layer	Minimum disclosure
Execution	Runtime substrate, sandboxing, filesystem access, network access, task timeout, step timeout, maximum steps, and evaluation entry point.
Tool	Tool list, schema style, tool-selection policy, error format, retryable error classes, and whether tools are model-visible or hidden middleware.
Context	Context cap, ordering policy, summarization/compression policy, retrieval method, persistent memory, and cache policy.
Scheduling	Agent loop, stopping rule, retry policy, escalation behavior, delegation policy, and rollback behavior.
Observability	Logged artifacts, trajectory format, checkpoints, traces, validation logs, and whether failures can be audited after the run.
Verification	Output parser, schema validation, self-checking, test execution, anomaly detection, and final patch validation.
Governance	Permission model, allowlist/denylist rules, side-effect boundaries, secret handling, and human approval points.

Table 4: Compact ETCSOVG disclosure example for the H_3 configuration in our controlled experiment.

Layer	Disclosure field	H_3 setting
Execution	Runtime and limits	Docker SWE-bench runner; 50 steps; 120s step timeout
Tool	Interface and errors	Task-relevant tools; minimal schema; structured errors
Context	Memory policy	32k token cap; summarize old steps; BM25 top-5 retrieval
Scheduling	Retry and recovery	3-attempt backoff; checkpoint rollback depth 3
Observability	Trace surface	Full trace logs; 10 retained checkpoints
Verification	Validation hooks	Self-checking; full output validation; anomaly detection
Governance	Permission model	Allowlist-oriented tool governance

B Harness Configuration Specifications

Model selection. The three models were selected before running the grid because they were tightly clustered on the LLM Stats coding leaderboard at selection time: on the April 23, 2026 snapshot, Kimi K2.6, GLM-5.1, and GPT-5.4 had coding scores of 45.4, 45.3, and 44.6, respectively.[3] The live leaderboard updates continuously as arena votes and benchmark columns change, so current readings may differ; the snapshot is used only to justify that the models were comparable when selected.

Subset construction. The controlled experiment uses SWE-bench Verified test tasks sampled into "subset100" with seed 42 and stratification by the SWE-bench difficulty label, preserving the approximate difficulty mix of the 500-task split (39 tasks $<$ 15 min, 52 tasks 15 min–1 hour, 8 tasks 1–4 hours, 1 task $>$ 4 hours). Each model-harness cell submits all 100 tasks in each of two final runs. Table 5 reports the subset composition; the ordered instance IDs are part of the experiment artifact.

Table 6 reports the model-side settings held fixed across harnesses. The experiment is intentionally a harness study rather than a decoding study: all three models are called through their official API services in their default configurations, preserving the provider-supplied model behavior as closely as possible. Each model uses the same endpoint configuration in H_1, H_2, and H_3, and no model receives harness-specific API tuning [27, 24, 44].

Before averaging the two final runs in Table 2, we inspect each run separately. Table 7 reports the two run-level 3×3 grids and recomputes HV and MV using the same population-variance convention as

[3] https://llm-stats.com/leaderboards/best-ai-for-coding, accessed April 23, 2026.

Table 5: SWE-bench Verified subset100 construction.

Field	Value
Source dataset	princeton-nlp/SWE-bench_Verified
Split	test
Full split size	500 instances
Subset size	100 instances
Sampling rule	stratified by difficulty
Random seed	42
$<$ 15 min fix	39
15 min–1 hour	52
1–4 hours	8
$>$ 4 hours	1

Table 6: Model API settings used in the controlled grid.

Model	Provider	Generation settings	Timeout
GPT-5.4	OpenAI	max output 4096	180s
Kimi K2.6	Moonshot	max output 4096	180s
GLM-5.1	ZAI	max output 4096	180s

the main table. The first run gives an HV/MV ratio of 8.72×, and the second gives 6.76×, so the harness-dominance pattern is not an artifact of averaging.

Table 8 expands the compact ETCSOVG card into the actual experimental specification. The first rows list shared controls that were intentionally held fixed; the remaining rows enumerate every harness mechanism varied in the final configurations. This is the level of detail needed to reproduce the variance decomposition and to determine whether a future comparison changes the model, the harness, or both.

Resource accounting complements the pass-rate results. Table 9 aggregates the selected final artifacts for all three models and all three harnesses, reported as if the runs had been executed sequentially. Sequential time is an accounting quantity, not the wall-clock time of a parallel campaign. In aggregate, H_2 is the cheapest controller in this implementation, while H_3 spends more on verification and recovery to obtain the best pass rate; per-model costs still vary with provider pricing and tokenization.

C Perturbation Stress-Test Protocol Details

The variance decomposition in Sec. 4.2 establishes that deliberate harness variation produces large HV. A complementary future test of the control-theoretic framing is whether agents exhibit the characteristic signature of closed-loop dynamical systems: disproportionate output instability in response to small controller-input perturbations. The following protocol follows directly from the theory; it has not been run, and we record it here for future work rather than as evidence for the thesis.

Setup. The planned experiment fixes GPT-5.4 and H_2 (the middle configuration from the variance decomposition experiment) and applies two classes of perturbation to the controller inputs. The first is context ordering: at each step, the top-k retrieved context chunks returned by the context construction function c are randomly shuffled before being passed to the model, so that the chunks themselves are unchanged and only their order varies. The second is tool output noise: numerical values in tool outputs are perturbed by a small multiplicative factor drawn from $\mathcal{N}(1, \epsilon^2)$ for $\epsilon \in \{0.01, 0.05, 0.10\}$, while text outputs and error codes are left unchanged. For each perturbation class, the protocol would run 50 independent trials per task on a 50-task subset of SWE-bench Verified, comparing them with 50 clean-baseline runs on the same tasks.

Metrics. We define the output instability score as the fraction of task trajectories whose final outputs differ from the clean-baseline outputs on that task, normalized by the perturbation magnitude where applicable. An agent that is robust to its controller inputs will produce instability scores close to the clean-baseline variance floor; an agent that is sensitive, as predicted by the dynamical-systems view, will produce instability scores that scale disproportionately with perturbation magnitude.

Table 7: Run-level pass@1 scores and variance decomposition before averaging across the two final runs. Score cells are pass@1 percentages on 100 tasks, with percent signs omitted. The highlighted HV column and MV rows are population variances over percentage-point scores, in pp^2.

	H_1 (Minimal)	H_2 (Improved)	H_3 (Full)	**HV**(M)
First final run				
GLM-5.1	52.0	57.0	66.0	**33.56**
GPT-5.4	55.0	59.0	64.0	**13.56**
Kimi K2.6	52.0	59.0	61.0	**14.89**
MV(H)	**2.00**	**0.89**	**4.22**	
Second final run				
GLM-5.1	53.0	56.0	65.0	**26.00**
GPT-5.4	55.0	58.0	63.0	**10.89**
Kimi K2.6	52.0	59.0	60.0	**12.67**
MV(H)	**1.56**	**1.56**	**4.22**	

First run: mean HV = 20.67, mean MV = 2.37, HV/MV = 8.72×. Second run: mean HV = 16.52, mean MV = 2.44, HV/MV = 6.76×.

Expected diagnostic value. This stress test would not be another benchmark score. It would measure whether the controller has enough damping to absorb small changes in context ordering and tool observations. If H_3 shows lower output instability than H_1 under matched perturbations, that would provide direct evidence that verification, drift checks, and rollback mechanisms improve closed-loop robustness rather than merely shifting the average pass rate.

D Industry Evidence Summary

The pattern argued for in Sec. 4.1 is even more visible in industry, where agent products compete on harness quality rather than model choice.

The managed agents architecture decouples the "Brain" (model and harness logic), "Hands" (execution environments), and "Session" (event log) into independent layers, reporting that harness assumptions go stale as models improve and that this decoupling reduced time-to-first-token by 60% at p50 and over 90% at p95 [20]. These are infrastructure improvements rather than model improvements.

The APEX-Agents benchmark, which evaluates agents on 480 real professional tasks across banking, consulting, and law, found that agent failures were predominantly execution and orchestration problems rather than knowledge failures [38]. Zero-score rates ranged from 40% to 62% across configurations using the same underlying model. The bottleneck was not what the model knew but how the harness managed execution.

The same pattern appears across public agent systems: SWE-agent exposes an agent-computer interface for repository navigation and editing [41], OpenHands packages software-development agents as an execution platform [39], and Codex CLI and Claude Code wrap frontier models in product-specific runtime layers [26, 2]; other public coding-agent systems including OpenCode [1], Terminus [10], and the Hermes Agent [31] make the same architectural choice.[4] These systems draw from a small set of frontier model providers [7, 35], yet they differ sharply in context handling, tool boundaries, execution isolation, and recovery behavior. The differentiating factor is the harness layer.

E Mechanism Analysis from Trajectory Logs

The factorial experiment in Sec. 4.2 reports aggregate variance and ranking statistics. This appendix records the trajectory-level mechanism analysis that supports the qualitative conclusions summarised in the main body.

[4] An open-source personal-AI variant is OpenClaw, described at https://openclaw.ai/.

Table 8: Full harness specification for the controlled experiment. The table includes shared runtime controls and harness mechanisms varied across H_1, H_2, and H_3.

Field	H_1 **Minimal**	H_2 **Improved**	H_3 **Full**
Source tasks	SWE-bench Verified subset100, seed 42, fixed task order	Same as H_1	Same as H_1
Runtime	Docker SWE-bench execution and evaluation pipeline	Same as H_1	Same as H_1
Step budget	50 agent steps; 120s per-step timeout	Same as H_1	Same as H_1
Design goal	Open-loop baseline with minimal intervention	Tool-robust closed-loop harness	H_2 plus self-checking and recovery controls
Context strategy	Append all prior steps chronologically	Compressed retrieval context	Same as H_2
Context cap	200k token estimate	32k token estimate	32k token estimate
History compression	None	Summarize older steps outside the recent window; compression window 8	Same as H_2
Retrieval	None	BM25 top-5 over older trajectory steps	Same as H_2
Context ordering	Chronological	Relevance then recency	Relevance then recency
Drift monitoring	Disabled	Disabled	KL-style drift check every 5 steps; threshold 2.0
Tool schema	Verbose tool descriptions	Minimal task-focused tool schema	Minimal task-focused tool schema
Tool exposure	All available tools	Task-relevant tool subset	Task-relevant tool subset
Tool error format	Raw tool errors	Structured error feedback	Structured error feedback
Runtime retry	No retry on tool failure	3-attempt exponential backoff	H_2 retry policy plus malformed-output retry
Retryable errors	None	timeout, rate limit, transient network	timeout, rate limit, transient network, malformed output
Failure escalation	No explicit recovery policy	Continue after recoverable tool failures and feed the error back as an observation	Checkpoint-style recovery on detected anomalies
Parser behavior	Basic parser with one tolerated malformed-output retry	Parser normalization for natural model outputs	Parser normalization plus validation layer
Output validation	Disabled	Schema-only validation	Full output validation
Empty patch handling	Allowed	Rejected when validation is active	Rejected when validation is active
Self-verification	Disabled	Disabled	Enabled; lightweight per-step self-check
Anomaly detection	Disabled	Disabled	Enabled for repeated action loops and context contradictions
Checkpointing	Disabled	Disabled	Enabled; retain last 10 state checkpoints
Rollback policy	None	None	Roll back up to 3 checkpoints on handled anomalies
Observability	Minimal logs	Full trajectory-level logs	Full trace logs with verification metadata
Governance	Permissive permission model	Permissive permission model	Allowlist-oriented tool governance

A subtle pattern in Table 2 also informs the leaderboard argument. Cross-model variance is smallest under H_2 (1.17 pp^2) and largest under H_3 (4.22 pp^2), with H_1 in between (1.72 pp^2). The H_1 vs H_2 comparison suggests that better-engineered closed-loop control can suppress model differences by absorbing variance through feedback; the H_3 result shows that some harness mechanisms (per-step verification, recovery) can also expose previously hidden model differences. This complicates the simple picture that better harnesses make model choice irrelevant: the relationship between controller quality and cross-model variance depends on which mechanisms the harness implements, which is itself part of why disclosure matters.

Mechanism analysis. The trajectory logs clarify the mechanism behind these numerical differences. The dominant $H_1 \rightarrow H_2$ pattern is reduced control noise rather than increased model knowledge: stricter JSON-only action formatting, smaller task-focused tool schemas, compressed recent-history context, and structured error feedback reduce malformed outputs, repeated exploration, and overly broad patches. In paired flip cases, H_1 often reaches a plausible local fix but fails because the trajectory exhausts its step budget, loses the relevant state in an append-only history, or introduces pass-to-pass regressions while trying to repair the target failure. H_2 more often keeps the model on a narrower action path: it retains recent task-relevant evidence, surfaces earlier relevant steps through retrieval, and makes tool / action boundaries clearer, producing smaller patches less likely to disturb retained tests. This is the qualitative counterpart of the variance result: H_2 does not make the model more knowledgeable, it makes the control loop better constrained.

Table 9: Resource accounting for the selected final artifacts, grouped by model. Sequential time sums per-cell elapsed time and does not assume parallel execution.

Model	Harness	Cost (USD)	Tokens	Sequential time
GPT-5.4				
	H_1 Minimal	\$34.19	18.9M	3.6h
	H_2 Improved	\$22.19	9.7M	2.0h
	H_3 Full	\$23.05	10.0M	2.4h
	Subtotal	\$79.43	38.5M	8.0h
Kimi K2.6				
	H_1 Minimal	\$24.29	60.4M	19.6h
	H_2 Improved	\$30.45	43.7M	11.0h
	H_3 Full	\$33.83	47.7M	20.7h
	Subtotal	\$88.57	151.8M	51.3h
GLM-5.1				
	H_1 Minimal	\$12.54	29.7M	9.2h
	H_2 Improved	\$13.86	16.5M	7.1h
	H_3 Full	\$21.39	24.5M	8.8h
	Subtotal	\$47.79	70.7M	25.1h
Total		**\$215.79**	**261.1M**	**84.4h**

The $H_2 \rightarrow H_3$ flips show a different mechanism. H_3 adds per-step verification, anomaly checks, and recovery signals, which turn false progress into visible feedback. H_2 sometimes accepts a patch that fixes the target test while breaking retained tests, or treats a failed command, empty test selection, or failed replacement as useful evidence. H_3 frequently spends more steps and tokens, but its verifier flags ineffective test commands, failed edits, incomplete fixes, and regression risks before submission. The additional H_3 gains therefore come primarily from closing the loop around verification and regression avoidance: the harness converts ambiguous execution traces into corrective observations that the same base model can act on.