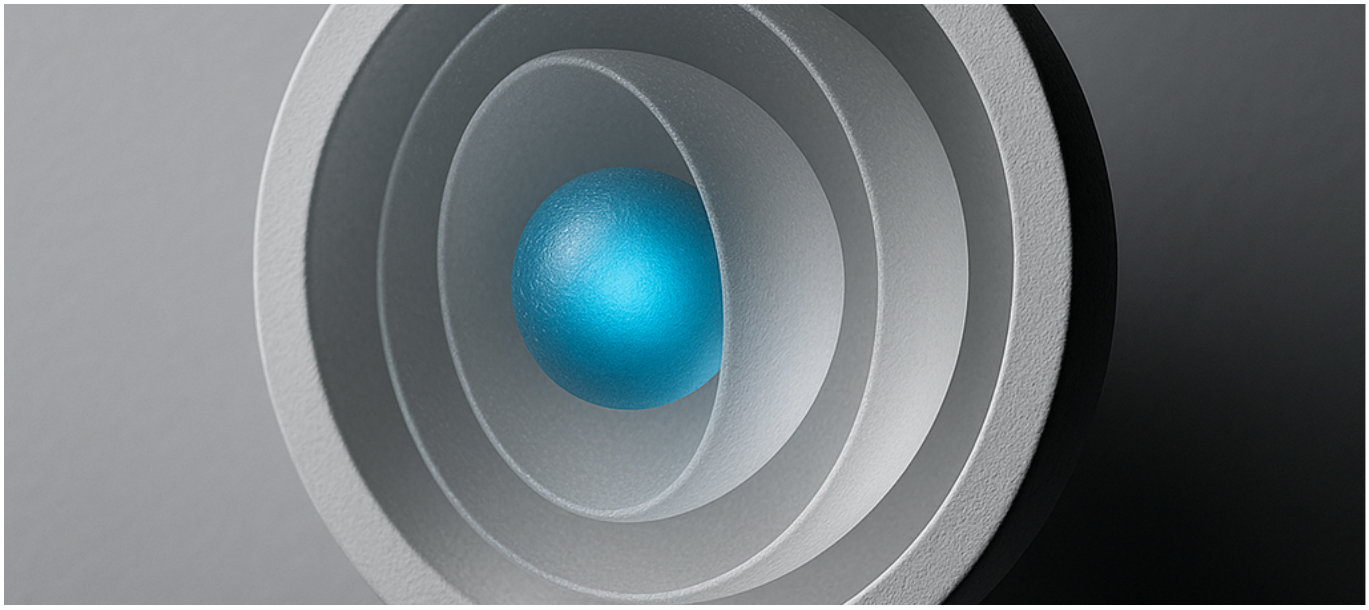

# The Harness, Explained: Why the Same Scaffold Scores 65% or 74% on SWE-bench

A 100-line Python harness scored 65% on SWE-bench Verified in July 2025. The same project now claims above 74% — and boots faster than Claude Code. The scaffold is doing work the weights get credit for.

## Why this matters now

Three things converged, and together they change how you should read every agent benchmark you see.

First, Anthropic published "Effective harnesses for long-running agents" on 26 November 2025 — an engineering post that treats the wrapper around the model as a first-class design artifact, not glue code.

Second, in its response to the NIST RFI on agentic security, Anthropic formally decomposed agent systems into four layers — model, harness, tools, environment — and argued that

agent security is a property of the whole system, not just the model. That is a vendor with a strong commercial interest in model quality saying, on the record, that model quality is one of four inputs.

Third, the regulatory clock. EU AI Act Article 14 human-oversight obligations bite for Annex III stand-alone high-risk systems on **2 December 2027**, and for Annex I safety-component systems on **2 August 2028**. Human oversight is not something a model has. It is something a harness implements — approval gates, logging, context for the reviewer. If you are building toward those dates, the layer you are least likely to have documented is the layer the regulation actually touches.

> Article 14 is a harness specification wearing a legal costume. No model weight satisfies a human-oversight duty.

## The concept: what a harness actually is

Anthropic's NIST submission gives the cleanest definition I have seen. The harness is:

> "the wrapper that lets a model run as an agent at all… [it] manages prompts, formats tool calls, carries context across turns, and runs the loop that keeps the model working until a task is done"

Strip that down: the model produces tokens. The harness decides what tokens it sees, what it does with the tokens that come back, when to stop, when to retry, and what survives between turns.

The analogy is the operating system. A CPU can execute instructions; it cannot run a program. Scheduling, memory management, I/O, process isolation — those are the OS. Swap one CPU for a faster one and your application performance moves a bit. Swap a well-tuned kernel for a naive one and it moves a lot more. Benchmarks that report "the model scored X" are reporting a CPU number while quietly holding an OS constant.

Anthropic's four layers sit in a clean hierarchy:

- **Model** — the weights, the reasoning capability.
- **Harness** — the loop, prompt assembly, tool-call formatting, context management.
- **Tools** — what the agent can invoke.
- **Environment** — sandboxing, filesystem scope, network egress policy.

Anthropic's argument about the bottom layer is the one worth memorising: the environment sets *"the hard boundaries"* regardless of what the model attempts. Everything above it is preference, persuasion, and probability. The environment is enforcement.

## How it works: the mechanics

### The benchmark evidence

The headline number comes from SWE-bench's own site: *"Jul 2025 mini-SWE-agent scores 65% on SWE-bench Verified in 100 lines of Python."* The framing on the benchmark's homepage is telling — it draws attention to the harness being minimal, not to the model being strong.

The mini-SWE-agent README now claims the harness is *"Performant: Scores >74% on SWE-bench verified; starts much faster than Claude Code,"* with Gemini 3 Pro reaching 74% under that scaffold.

Two things follow. One: a hundred lines of Python is enough scaffold to reach the top tier of a benchmark that whole companies build products around. Two: benchmark deltas across leaderboard entries are model-plus-harness deltas, and nobody publishes an ablation separating them.

> Every agent benchmark number is a pair. You are shown one half of it and invited to attribute the result to the other.

## METR is explicit about this

METR defines the 50% time horizon as the task length — measured in human expert time — at which a model succeeds half the time. Critically, METR ties the metric explicitly to its own tools-and-environment setup. The metric measures a model-plus-harness pair. METR says so.

The Frontier Risk Report covering February–March 2026 puts the public frontier 50% time horizon at roughly **12 hours**, with a range spanning **5 to 61 hours**, and notes that the best measured 50% time horizon of any shared model was lower than 20 hours. A 5-to-61-hour spread is not noise you can wave away — it is an order of magnitude, on a metric that is definitionally harness-dependent. "The model can now do X-hour tasks" means it can do them *in this scaffold*.

## Durability: what separates toy loops from long-running agents

Anthropic's engineering post describes a two-agent pattern for long-running work: an **initializer agent** that runs exactly once, and a **coding agent** that runs every subsequent session.

The interesting design choice is where state lives. The harness enforces durability through on-disk artifacts rather than context:

- `init.sh` — a boot script
- a git repository with an initial commit
- `feature-list.json`
- `claude-progress.txt` — a log that survives context resets

That is a storage design decision, not a prompting trick. Context windows are volatile memory; the filesystem is durable storage. A harness that keeps plan state in context has an agent that forgets its plan the moment the window compacts. A harness that writes to disk has an agent that can be killed and resumed.

> If your agent's plan lives only in the context window, you do not have a long-running agent. You have a long prompt.

## Where oversight and permissions get implemented

Every credible agent-security framework published in the last year describes controls that live in the harness and environment layers, not the model.

Anthropic's framework for developing safe and trustworthy agents describes a coding agent with read-only permissions by default: it may analyse files without approval, but must request explicit human approval before any action that modifies code or systems. That is a harness-enforced gate on a tool-layer capability.

Google's agent security principles set out three requirements — well-defined human controllers, limited agent powers under least privilege, and observable, logged actions — plus a two-layer defence: deterministic runtime policy enforcement using action manifests *before* execution, with reasoning-based defences layered on top. The ordering matters. Deterministic first, model judgement second.

OWASP LLM06:2025 Excessive Agency recommends human-in-the-loop control for high-impact actions, with approvals logged and auditable and humans given sufficient context — and explicitly names rubber-stamping as the failure mode. That last clause is the one teams skip. An approval dialog with no context is a compliance artifact, not oversight.

LangChain-ecosystem production hardening guidance maps agent design onto OWASP LLM06 and LLM02:2025 Sensitive Information Disclosure, recommending narrow single-purpose tools over generic shell or HTTP tools, and separate database roles for read versus write. Again: tool and environment design, not model selection.

And MiniScope (arXiv 2512.11147) proposes a formal least-privilege framework for authorizing tool-calling agents, reconstructing permission hierarchies that reflect relationships among tool calls — moving from engineering heuristics to checkable access-control logic. Permissions as something you can verify, not something you hope the system prompt covers.

The pattern is visible in production too. Cloudflare's open-sourced internal AI workspace starts agents with zero access and grants only task-specific permissions — the environment layer doing the work.

# Trade-offs and limits

## Minimal harnesses win benchmarks; benchmarks are not your workload

SWE-bench Verified tasks are well-scoped: a repository, a failing test, a bounded edit. A 100-line loop is sufficient because the environment does the constraining. Your production agent does not get a failing test as a success oracle.

So the generalisation from "65% with 100 lines" to "you don't need a framework" does not hold. What holds is narrower: *on tasks with crisp verification and a bounded filesystem, harness complexity has sharply diminishing returns*. Whether your task has those properties is the question the benchmark cannot answer for you.

## Benchmark validity is itself contested

NeurIPS 2025's *"Establishing Best Practices in Building Rigorous Agentic Benchmarks"* calls for detecting outcome-validity issues such as grading errors, and for explicit reporting when validity cannot be guaranteed. Read that alongside the harness problem: we are comparing model-plus-harness pairs on benchmarks whose grading correctness is an open research topic.

## Nobody is documenting any of this

The *2025 AI Agent Index* found that **25 of 30** documented agents disclose no internal safety results, and **23 of 30** report no third-party testing. If five-sixths of shipped agents publish nothing about their own safety evaluation, the harness layer — the least glamorous of the four — is certainly not being documented either.

That is a supply-chain problem for anyone buying agentic products. You can read a model card. There is no harness card.

## The oversight tax is real and mostly unpriced

Read-only-by-default with explicit approval before mutation means a human in the loop on every write. Deterministic policy enforcement before execution means maintaining action

manifests. Narrow single-purpose tools instead of a generic shell means writing and maintaining many small tools instead of one large one.

Every one of those controls costs latency, engineering time, and autonomy. The Article 14 deadlines sound distant until you consider that the controls are architectural. You do not bolt human oversight onto an agent designed to run unsupervised; you rebuild the loop.

### Portability is not what you think

If a harness contributes materially to benchmark scores, harness behaviour is model-specific. A scaffold tuned around one model's tool-call formatting, error recovery, and context-compaction behaviour is not guaranteed to transfer. Swapping models under a fixed harness is a change to the pair, and the pair is what you measured. That is my reading, not something the research establishes with an ablation — but the 5–61 hour METR spread is consistent with it.

## My take

**My take:** the four-layer decomposition is the most useful thing to come out of the agent-security discourse this year, and it is useful precisely because it is boring. It gives you four places to look instead of one place to argue about.

I read Anthropic's NIST framing as an admission with commercial consequences. A model vendor stating that security is a property of the whole system is telling enterprise buyers that model choice is not sufficient — honest, and inconvenient for anyone selling model access as a safety story.

I expect harness disclosure to become a procurement requirement before it becomes a regulatory one. The 25-of-30 and 23-of-30 numbers describe a market where nobody discloses because nobody asks. The first large enterprise that asks "which harness, which tools, what environment boundaries, what approval gates" in an RFP will change that faster than any 2027 deadline.

I also expect the MiniScope direction — formal, checkable permission hierarchies over tool calls — to matter more than better prompting. Google's ordering is the tell: deterministic

runtime enforcement first, reasoning-based defences on top. Anything you can enforce in code, you should enforce in code, because the environment sets the hard boundaries and the model sets none.

Two practical recommendations:

- **Write your harness down.** Not the code — the contract. What the loop does, what survives a context reset, what requires approval, what is logged. If you cannot produce that document, you cannot demonstrate Article 14 compliance and you cannot debug your agent's failures.
- **Treat the environment as the security boundary and everything above it as best-effort.** Filesystem scope, network egress, database roles split read from write. The state of authentication in the MCP server ecosystem is a useful reminder of what happens when the tool layer is trusted by default.

One honest uncertainty: I have no ablation showing how much of the 65%-to-74% movement in mini-SWE-agent is harness improvement versus model improvement — the README attributes the 74% figure to Gemini 3 Pro under that scaffold, and the 65% figure is from July 2025. Both numbers are real; the decomposition between them is not published. That gap is exactly the problem this article is about.

**Stop asking which model your agent uses and start asking which harness runs it — the benchmark number you were shown belongs to the pair, and only one half of it is on the invoice.**

WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting 