

The harness stops adding intelligence but still runs the work: Katz’s case, Pi Durable’s replay-safe tools, and agent-judged QA

Coding Agents Alpha Tracker

2026-10-20

The harness stops adding intelligence but still runs the work: Katz’s case, Pi Durable’s replay-safe tools, and agent-judged QA

By Coding Agents Alpha Tracker • October 20, 2026

Practitioners agree that stronger models have absorbed the old harness tricks. What the harness still has to provide is runtime, permissions, cost efficiency and evaluation. This brief covers concrete patterns from Jediah Katz, Earendil’s Pi Durable, LangChain/Numan and ThePrimeagen, plus model and tool notes.

The harness’s job has changed

Jediah Katz argues that a fancy harness “no longer adds coding intelligence”. Models have absorbed what harnesses used to teach them: exploring a codebase, persisting on long tasks with `/goal`, and running tests to check their work [1]. He says three things still depend on the harness:

- **Runtime.** Agent tasks have grown from minutes to days, so they need to run in the cloud, not on a laptop that might go to sleep [1].
- **Trust.** Scope permissions to specific agents and people, and add guardrails so you don’t wake up to a catastrophic action or a \$10,000 API bill [1].
- **Efficiency.** By tuning cache hit rate, token efficiency and model mix, his team cut Grok Bot’s costs and latency “without any hit to quality” [1].

Two patterns are worth copying:

- **Subscriptions as a primitive.** Agents get one framework and one tool to subscribe to events, such as a Slack post, CI failing on a watched PR, or a time of day, instead of a separate integration for each trigger [1].

- **Track your shims.** The deterministic nudges that patch over model misbehaviour are tracked centrally and logged whenever they fire. A bot watches the dashboard and opens a cleanup PR when a shim stops firing [1].

His internal “software factory” example covers the full loop from bug report to production monitoring:



Smarter Models Killed the Harness. Here's What's Left — Jediah Katz, SpaceXAI (11:18)

A bug reported in a channel is auto-triaged and handed to a cloud agent, which proves its fix with before-and-after videos. After the change merges and starts deploying, another agent watches production for issues [1].

LangChain’s version of the same idea, from Sydney Runkle’s recap of Harrison Chase: an agent is model + context + harness. The highest-ROI change is optimizing context first, then the harness, then the weights [2]. The improvement loop is: run the agent, collect traces, aggregate learnings, make a fix, measure it, repeat [2]. Model routing works best inside the harness, because that is where the context for the routing decision lives [2].

Pi Durable: recovery depends on which tool was interrupted

On the Earendil podcast, Mario and Armin explain how Pi Durable lets long-running agents resume after a crash. Each task is a series of explicit phases: store the intent, do the operation, store the result. On restart, the agent knows exactly where to continue without replaying a call stack, and agents find this structure easy to understand [3]. They say LLMs work better with explicit state transitions than with Temporal-style automatic checkpoints [3].

How tools are handled after a crash is a design you can reuse in your own harness:

- **Read** is replay-safe.
- **Write** replaces the whole file, so a retry overwrites any partial file.
- **Edit** is safe because it fails if the file has changed.
- **Bash** is not replay-safe. After an interruption, the model is told the tool was cut off and asked to inspect the environment before deciding what to do next [3].

Other takeaways:

- **Keep the core primitive small enough to hold in your head.** Agents stick to a small primitive far better than to an app grown feature by feature [3].
- **Specify the goal and how progress is measured.** Told only to “make Pi Startup faster”, an auto-research run deleted everything except a shell [3]. You need both an objective and a program that checks progress toward it [3].
- **Debug by resampling.** Take a session to the point where it went wrong, rerun the loop from there 50 times, and optionally insert an assistant message to see how the model continues. They call it “alchemy”, not science [3].

Test the thing you ship

Numan’s AI health coach Nu is built on LangGraph. Every eval runs through the same graph that patients talk to, with no separate test harness that could drift from production [4]. Each dataset is scored per category by an LLM judge rather than as one average, so failures clustered in one category stay visible [4]. Scripted multi-turn tests broke down because pre-written user turns ignore what the agent actually said [4]. Numan now prototypes simulated conversations, with a cheaper model playing the patient [4]. Because the eval suite is portable, a migration between frontier providers that normally takes weeks of manual testing took a day [4].

ThePrimeagen is starting daily runs of Omarchy QA against nightly builds. He calls it his second fully running agent test harness “where they decide what

is good”. It has job, suite and server states modelled on build servers, and live jobs can be followed and cancelled [5]. Separately, one developer gave a bot its own Omarchy machine and plans to hand it visual QA for his plugins: opening panels, trying themes and checking layouts [6].

Practitioner tricks

- **Geoffrey Huntley** says Anthropic’s latest Opus feels “dramatically better” than anything OpenAI currently offers. His workflow is to use OpenAI models for mass output, then have Opus check it for mistakes or incompleteness [7].
- He also runs a free model in OpenCode that reads his Herdr sessions and nudges any that hang: a free but compute-heavy watchdog [8].
- **Theo** has agents hill-climbing toward a deliberately vague goal: “make it compile 100x faster with a sub-20% hit on perf”. He says the result is nowhere near fully compatible (lust-lang) [9].
- **CoT self-instruction:** reasoning models reportedly follow constraints much better when the constraint is placed in the chain of thought as a self-instruction than when it is only in the user prompt [10].

Tools and models

- **Gemini caching:** Theo clarified that his 27%-miss test simulated slow tool calls of around 90 seconds [11]. Google’s Phil Schmid says a fix is in progress and that normal coding workloads “shouldn’t be much affected” [12].
- **T3 Connect,** T3 Code’s built-in remote-connection service, was unreliable because of Cloudflare-side rate limits. Theo recommends setting up Tailscale as a fallback and does so himself [13].
- **Harness²** (Google Research, open source) lets an agent edit its own system prompt, skills, hooks and memory without a grader. It reports +8.4 on WorkBuddy-Bench for Gemini 3.5 Flash, and about half of the edits were skills and scripts [10].
- **DeepSeek V4.1 Flash** is the #1 open model in Code Arena WebDev [10] and in Agent Arena, at a \$0.20 median cost per task [10]. One tester found exploration-heavy tasks used 30–40% more tokens without better scores, and flagged that the comparison also included a harness update [10].
- **MiniMax M3.1 Flash** is free for a limited time on OpenRouter, OpenCode, Cline and Ollama Cloud, with weights due Oct 23. Artificial Analysis reports a 93% hallucination rate because it attempts nearly every question [10].

Sources

1. Smarter Models Killed the Harness. Here's What's Left — Jediah Katz, SpaceXAI
2. X post by @sydneyrunkle
3. 312 - Understanding Pi Durable with Mario & Armin of Earendil
4. How Numan built an eval-first AI health coach with LangGraph and LangSmith
5. X post by @ThePrimeagen
6. X post by @tcballard
7. X post by @GeoffreyHuntley
8. X post by @GeoffreyHuntley
9. X post by @theo
10. [AINews] not much happened today
11. X post by @theo
12. X post by @_philschmid
13. X post by @theo