Harnesses, Demo Loops, and Routing Rules

Coding Agents Alpha Tracker

2026-07-06

Harnesses, Demo Loops, and Routing Rules

By Coding Agents Alpha Tracker • July 6, 2026

Today's strongest coding-agent signal: performance is increasingly a harness problem, not just a model problem. This brief covers Simon Willison's agent-recorded demos, Theo's concrete routing playbook, Addy Osmani's eval/context framing, and the new llm-coding-agent release.

TOP SIGNAL

The biggest edge right now is moving from **"pick the best model"** to **"build the right harness"**. Addy Osmani says a coding agent is *model + harness*, with the harness doing the heavy lifting—sandboxes, tool permissions, memory, observability, and evals—and points to a Terminal Bench 2.0 jump from outside the top 30 to top 5 **without changing the underlying model** [1]. Theo's production setup is the same idea in operator form: a `Claude MD` glossary for "intelligence" and "taste", explicit routing across Fable/Codex/Opus, and an escalation rule to *judge the output, not the price tag* [2].

TRY THIS

- **Turn `--help` into an agent skill, then require a demo.** Simon Willison's new `shot-scraper video` flow lets an agent record a browser demo from `storyboard.yml` using Playwright [3]. His exact move: point the model at the branch, tell it to run `uv run shot-scraper video --help`, then have it record the feature against a local Datasette instance and demo DB; in his example, the storyboard YAML itself was constructed entirely by GPT-5.5 xhigh in Codex Desktop [3]. Simon's reusable pattern: rich `--help` output can function like a built-in `SKILL.md` for agents [3].

  ```
  Review the changes on this branch.
  cd to ~/dev/shot-scraper and run "uv run shot-scraper
  video --help" [3]
  ```

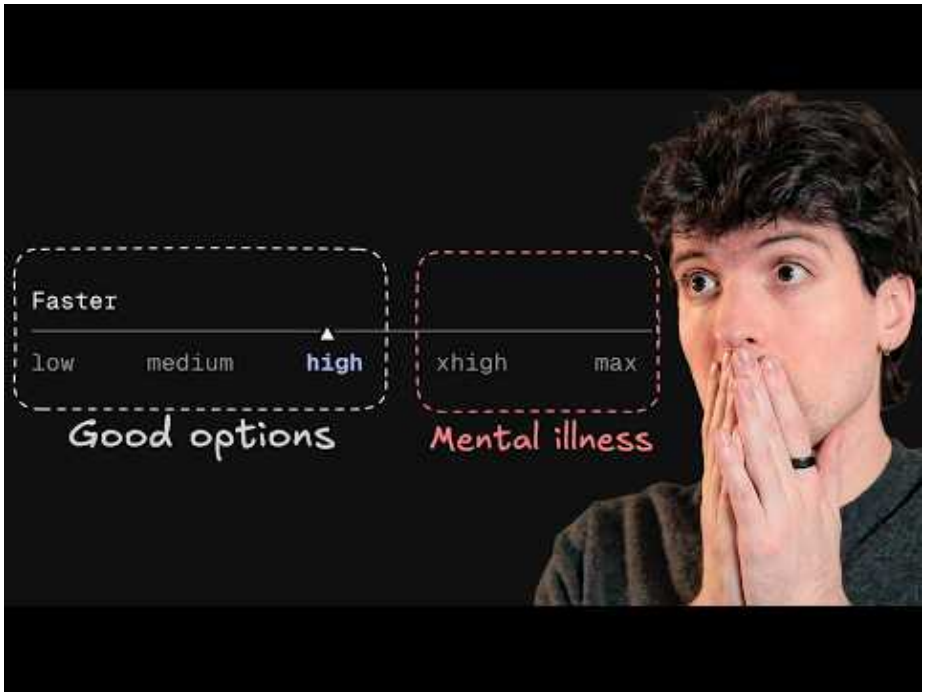
- **Write your routing policy in plain English.** Theo's `Claude MD` pattern is concrete: define what **intelligence** means (how hard a task the model can handle unsupervised) and what **taste** means (UI/UX/API/copy quality), then set defaults—Fable for best intelligence/taste, Codex 5.5 for high intelligence at low cost, Opus 4.8 for high taste [2]. Add the override rule verbatim: *judge the output, not the price tag* [2]. His practical warning: keep Fable at `high`; he says `X-High/Max/Ultra` tend to overthink and inflate bills without improving results [2].
- **Pick the control mode before you pick the model.** Cat says hard tasks often go better with **one agent** so you can quickly correct bad assumptions and track progress, even if you usually run tens of agents in parallel [4]. For broad async work, she asks Claude to write regular summaries of all running agents "like a chief of staff" and keeps tightening the format until the report is denser and more actionable [5]. Addy's vocabulary is useful here: *conductor* mode for hands-on IDE steering on the tricky 20%, *orchestrator* mode for async swarms on broad objectives [1].
- **Add an eval lane next to tests.** Addy distinguishes deterministic tests from evals that inspect the whole trajectory: a secondary constrained LLM judge can fail a build for destructive commands, leaked keys, or pulling in unvetted libraries even if final tests are green [1]. Theo's rule of thumb points the same way: review a much smaller percentage of raw code than you did five years ago, and if code is critical enough for hand review, generate a lot of verification code on top of that human pass [6, 7].

WHAT SHIPPED

- **shot-scraper 1.10** — adds `shot-scraper video`, which takes a `storyboard.yml` routine and uses Playwright to record a browser demo. Start with the video docs and the repo [3].
- **llm-coding-agent 0.1a0** — compact coding agent on Simon's `llm` framework with six tools: `edit_file`, `execute_command`, `list_files`, `read_file`, `search_files`, `write_file`. Run it with `uvx --prerelease=allow --with llm-coding-agent llm code`, then read the spec and commit sequence [3].
- **Sandbox signal:** Peter Steinberger says he can't recommend crabbox.sh enough as a way to use @useblacksmith for agent sandboxes [8, 9].
- **Current practitioner routing snapshot from Theo:** Fable for intelligence + taste, Codex 5.5 for bulk mechanical work/data analysis/migrations and computer use, Opus 4.8 for high-taste review. His reported workload: 11-12 merged PRs from one thread in 2-3 days, at roughly $150 total across Fable and helper models [2].

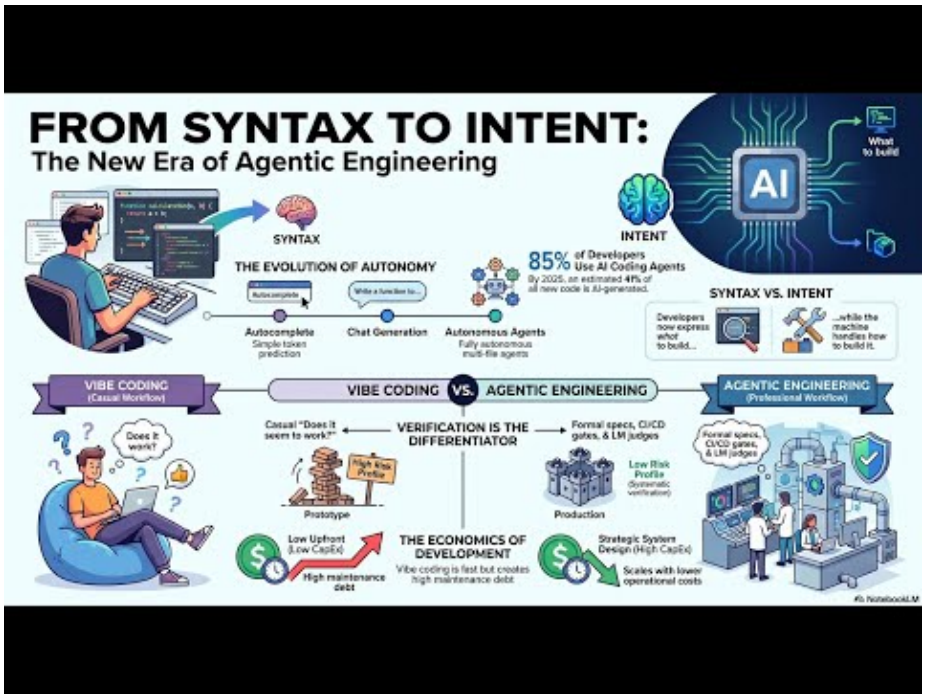
GO DEEPER

- **16:49-19:09 — Theo on model-routing vocabulary.** Good clip if you want a reusable `Claude MD` rubric instead of hand-wavy "use X for Y" advice: define intelligence, define taste, set defaults, then escalate when the cheap model misses the bar [2].



A proper guide to Fable 5 (16:49)

- **6:12-8:05 — Addy on tests vs evals.** Clean explanation of why a green test suite is not enough for autonomous agents: the eval judges the **path**, not just the final state [1].



(Podcast) The Vibe Coding Revolution and the Rise of Agentic Engineering (6:12)

- **shot-scraper** + video docs — worth studying if you want agents to prove UI work with reproducible browser recordings instead of screenshots and trust-me text [3].
- **llm-coding-agent** + spec — a compact reference implementation for a code-editing tool surface on top of Simon's `llm` framework [3].

Editorial take: the alpha is shifting from "which model won this week?" to "what routing rules, context scaffolding, and proof loop did you build around it?" [1, 2, 3]

Sources

1. (Podcast) The Vibe Coding Revolution and the Rise of Agentic Engineering
2. A proper guide to Fable 5
3. Have your agent record video demos of its work with shot-scraper video
4. X post by @_catwu
5. X post by @_catwu
6. X post by @theo
7. X post by @theo
8. X post by @cdxker

9. X post by @steipete