Make agent corrections enforceable, and choose Opus or Fable by whether the work can be verified

Coding Agents Alpha Tracker

2026-11-13

Make agent corrections enforceable, and choose Opus or Fable by whether the work can be verified

By Coding Agents Alpha Tracker • November 13, 2026

Practitioners at AIE Code explained how they make agent output trustworthy, and Theo gave a working rule for choosing between Opus 5.5 and Fable 5.2. Gemini 4 Hydrogen launched as Google's fast, cheaper model, and early reports on it are mixed.

Turn agent corrections into checks the system enforces

The most useful AIE Code material came from @poteto (Lauren), in notes TheTuringPost posted and swyx shared. They describe how her team made coding agents trustworthy:

1. **Clean the code agents copy from.** Agents copy whatever looks like an established pattern, including workarounds and deprecated code [1].
2. **Make each correction enforceable.** Each time you correct an agent, ask whether the fix can become a lint rule, type constraint, test or better API. A growing AGENTS.md still depends on the agent following every instruction [1]. Her team limits React's `useEffect` to framework internals because agents kept misusing it, and it enforces import boundaries between processes [1].
3. **Give agents tools to inspect the product.** That means debugging tools, reusable CLIs, and a "feature map" that describes the app from the user's side. Then ask agents for screenshots, videos or state diagrams rather than written explanations [1].

4. **Spend heavily on verification.** The team uses runtime verification and is trying property-based UI testing and TLA+. Most of her tokens go to verification [1]. Agents now test releases, find bugs, open fixes and merge them automatically, but building enough trust for that took about six months [1].

Reported results: 11,000 PRs in a month with a revert rate around 0.5% [1].

Kent C. Dodds's friction log. On ThursdAI, Dodds described three goals for a software factory: produce good work, catch mistakes, and limit the damage when one gets through [2]. He also makes agents report what was painful in the codebase, then has another agent fix those problems [2]. The setup is a Kody automation with a very short skill that tells the agent to run one CLI command, which returns the next step. The steps are: mark the PR ready, wait for CI and agent review, fix errors, merge when everything passes, then report what was bad about the process [2].

incident.io's numbers. These come from The Pragmatic Engineer. Usage took off once engineers ran several agent sessions in separate worktrees, and nearly 80% of production PRs now include agent-generated code [3]. Spend fell from about $4,500 to about $1,600 per engineer per month after the move from Opus 5 to Opus 5.5 [3]. About 80% of that goes to Claude and 20% to Cursor [3]. PRs per engineer rose only about 10% at the median but about 70% for tenured engineers. New hires hold off on AI for their first month [3]. Most incidents now come from missing information about production rather than bad code, so the team is building ways for agents to observe production [3].

Theo: Opus 5.5 for work you can verify, Fable 5.2 for judgment calls

In "Stop Picking The Wrong Model," Theo agrees with starting on Opus 5.5 at medium effort but prefers high. He says medium ends its turn too soon, and he avoids Max [4]. He uses xhigh for tricky work that needs a lot of testing, such as Windows bugs or replaying large logs [4]. Use Haiku for bulk pattern-matching across files, not for writing code [4].

- **Prompt tip:** Opus rarely questions the task it's given. Adding "If you think this is a good idea, do it. Otherwise, suggest something else" meaningfully improved his results [4].
- **His rule:** if the work can be verified and you can iterate toward it, Opus is enough. For work that needs creativity or exploration, or when Opus keeps going in circles and you need a call on whether to ship, use Fable [4]. Fable is slightly more likely to let issues slip in review [4].
- **Worked example:** Opus built a test environment for performance problems on very long T3 Code threads, but missed the key bug. Theo then had Fable verify that setup and look for the cases that broke. It found that unresolved to-do lists pulled in the whole thread history, and proposed sending the goal as a separate JSON field. That cut data over the

wire about 150x in some cases [4].

Separately, Theo turned his worst cases (bad mobile internet plus very long threads) into a measurable target to improve against, using fake devices and iOS simulators [5].

Gemini 4 Hydrogen: cheaper and faster, with mixed early reports

Hydrogen scores 54.3% on CursorBench at default effort, against 39.6% for Gemini 3.8 Flash. It costs $1.50/M input and $7.50/M output tokens (evals) [6, 7]. Antigravity reports 57% on Terminal-Bench 4.1 [8].

Antigravity also added **Orchestrator**, which splits tasks between Carbon and Hydrogen. Google says it uses 24% less quota and finishes the same tasks 19% faster [9]. Nicholas Moy's advice for Ultra users is to use Hydrogen for quick tasks and save Carbon tokens for high-level work [10]. Google's Ivan Leo prototypes with Hydrogen because it can call several tools at once. He sends long investigations across the codebase to Carbon. In one case, Carbon reproduced 400 errors, traced them to a permissions change in a merged PR, and drafted a message to the person who merged it [2].

The pushback: - **Pricing.** Simon Willison points out that Hydrogen is only 25% cheaper than Argon and Carbon. He also found its pelican SVG drawings notably worse [11]. - **Speed.** Theo calls its speed "garbage" next to Sol and Opus, even with those running at regular tiers rather than fast ones [12].

Also worth knowing

- **Modal's sandbox design:** Erik Bernhardsson argues that a harness should run for a long time outside short-lived sandboxes, possibly one per tool call. That design makes it easy to spawn subagents, snapshot work and resume it [13]. Modal doesn't mandate a coding tool. Engineers moved from Cursor to Claude and Codex, and some are trying OpenCode [13].
- **ADEMark v1** benchmarks coding tools against a bare CLI using the same Mac mini, tasks and model. Most tools used fewer tokens than the bare CLI, and one sat at 16% CPU while idle [14].
- **Claude Meetings** adds Claude to video calls so it can build docs and whiteboards during the call. It's rolling out in beta on Team and Enterprise plans [15]. Alex Albert presents it as a way to give Claude context it would otherwise be missing [16].
- **OpenAI** published its practices for securing persistent agents [17]. It also announced two cyber products, Daybreak Defense Factory and Daybreak Red Team [18].

Sources

1. X post by @TheTuringPost
2. ThursdAI Nov 12 - No more Jensen at Nvidia, Gemini 4 Hydrogen, Fable 5.2, software factories & more
3. The Pulse: Details of incident.io's AI spend
4. I think I figured it out
5. X post by @theo
6. X post by @cursor_ai
7. X post by @cursor_ai
8. X post by @antigravity
9. X post by @_mohansolo
10. X post by @thenickmoy
11. X post by @simonw
12. X post by @theo
13. Fireside Chat with Erik Bernhardsson, Modal
14. X post by @zqkrq
15. X post by @claudeai
16. X post by @alexalbert___
17. X post by @kevins8
18. X post by @gdb