

Theo's repo audits find Gemini 4 Argon wrong on most claims, as Osmani lays out the verification stack agents need

Coding Agents Alpha Tracker

2026-10-15

Theo's repo audits find Gemini 4 Argon wrong on most claims, as Osmani lays out the verification stack agents need

By Coding Agents Alpha Tracker • October 15, 2026

Theo tested Gemini 4 Argon by checking every claim it made against his real codebases, and the results were poor. Addy Osmani and LangChain describe the property tests, differential tests and evals that let agents check their own work. Also: tooling for running many agents in parallel, Huntley's "remix" model for open source, and Ben Tossell's AGENTS.md habits.

Test new models on your own repo: Theo finds Gemini 4 Argon wrong on most claims

Theo says Gemini 4 Argon is "outright unusable for real code work in my codebases" because "it aggressively hallucinates about how systems work" [1]. In almost every test he ran, more than half of its claims were wrong [2].

His method is worth copying more than his verdict:

1. Have each model audit a real codebase.
2. Check every claim it makes.
3. Have a panel of judges score the results [3].

On his ts-rust audit, GPT-6.1 Sol made 38 claims, all true. Argon scored 4.38, and 24 of its 36 claims were false [3]. On his Lakebed audit, 56 of Argon's 78 findings were wrong [3]. He notes this contradicts Artificial Analysis's low hallucination rate for the model [3]. His advice: if you do use Argon, run its output through Sol or Opus [3].

The SQLite case shows the failure pattern and how to recover:

- Argon’s diagnosis was mostly right, but its PR was about 1,200 lines for a change Theo says should have been about 30. It ignored the repo’s PR-format skills, failed CI and lint, and left review-bot comments unaddressed [3].
- He handed the thread to Opus 5.5 for a second-agent review. Opus found that Argon fixed the wrong half of its own diagnosis and missed two existing PRs that already solved the problem [3].
- The final fix was 21 lines and has been merged [3].

For frontend work, Theo recommends Anthropic’s official design skill but not its taste skill [3]. He prefers fast models he can iterate with over expecting a good first attempt [3].

Salvatore Sanfilippo (in Italian) says Argon trails Opus 5, GPT-6.1 Sol and GPT Astra in agentic coding [4]. He also says coding works well at around 250k–350k tokens of context [4].

Osmani: E2E, property and differential tests the agent can run

The last brief covered Addy Osmani’s “define done” advice. His new post adds the specific kinds of test to invest in. His premise is that the bottleneck is now verifying code, not writing it, and he would “always prefer a much better feedback loop to a slightly better model” [5].

- **E2E tests as ground truth.** Keep a small, high-value set [5]. Failures should report which step failed, with screenshots and traces, and “Make this E2E test pass” gives the agent a clear definition of done [5]. Via poteto: give the agent a maintained skill or CLI to launch, drive and observe the app, plus a user-POV feature map [5].
- **Property tests** (fast-check, Hypothesis, proptest). These stop the agent from passing only test examples it picked itself [5]. Start with a pure function, write the property in plain English, save minimized failures as regression cases, and ask the agent for properties rather than examples [5].
- **Differential tests for migrations.** Treat the old system as the oracle. Feed both systems random inputs or replayed live traffic [5], normalize irrelevant differences, and keep a list of approved changes [5].
- **Fast, deterministic loops.** Slow or flaky tests teach the agent to retry rather than fix [5]. Aim for a subsecond core loop with seeded randomness, frozen clocks and isolated flaky tests [5].

LangChain’s evals FAQ makes the same point for agents in general. To evaluate a coding agent, run the repo’s tests after it finishes, with a clean copy of the repo for every run. Its open-source Harbor harness sets this up [6]. Start with 10–20 examples taken from traces, save a baseline, and add each new failure as

a case [6]. Use narrow, binary questions for LLM judges [6]. LangChain’s Eval Engineering Skill runs in Claude Code or Codex and helps write evals for your repo [6].

LangChain also explained how its Deep Agents harness (`uv add deepagents`) manages context:

- Tool results over about 20k tokens are saved to a file, and the model sees a preview plus the file path [7].
- At 85% context use, older messages are summarized into a file the agent can read back [7].
- Memory is a Markdown file you own, so it carries over if you switch providers [7].

Tooling for many agents at once

Anthropic says it went from about 2,500 merged changes a day in May to more than 30,000 in September. Infrastructure built for humans became the bottleneck, so it built a new version control system and moved its monorepo off git in five months [8].

At a smaller scale, **Kent C. Dodds**’ Kody now queues git pushes in a durable object. When 20 agents pushed to the same branch at once, some had to rebase and resolve conflicts, but all changes landed [9].

The **Claude Code desktop app** now starts sessions faster, makes it easier to split work across sessions, and shows what each one is doing [10]. Theo says this validates UX experiments T3 Code tried first [11]. Claude Platform traces from the Messages API, Managed Agents and the SDKs can now go to LangSmith over OpenTelemetry, in public beta (docs) [12].

Huntley: open source that accepts remixes, not PRs

Geoffrey Huntley is closing pull requests on Kittens but keeping issues and discussions open. To change it, you “point your agent at it” and patch it with Nix overlays [13]. In his “remix model”, a small overlay describes the change once and is reused across devenv shells, Docker images, NixOS tests, CI and agent sandboxes, with no fork to maintain [14]. His example is an overlay that makes Git refuse force pushes [14]. He suggests running agents over GitHub’s network graph to summarize what people are remixing [14].

Tossell’s AGENTS.md habits

From **Ben Tossell**’s Stanford GSB talk:

- He puts “Questions are requests for an answer, not for changes” in his AGENTS.md, so a question doesn’t trigger edits [15].
- He routes task-specific context (travel, his fund) to separate files [15].

- Start with one agent, and promote anything you keep repeating into the instructions [15].
- His instructions ask for three distinct static mockups for anything he builds [15].
- When the agent misses, he adds context step by step: screenshots, voice rambles, then screen recordings or annotated tldraw screenshots [15].

Smaller notes

- **Theo** logged all his Claude and Codex requests to a ClickHouse database [16]. The data shows newer models increasingly running Python instead of bash or other tools [17], and his own near-complete move from Codex to Claude [18]. He also points out that Codex’s system prompt still doesn’t name the current model, only “based on GPT-6” [19].
- **T3 Code** has 540k users [20]. Nightly adds SQLite performance fixes meant to stop the “reconnecting” banners [21].
- **Romain Huet** (OpenAI) says GPT-6.1 Sol Ultrafast is up to 8x faster than standard mode [22].
- **Greptile** has committed \$150,000 in tokens for agent PR reviews across all Omarchy repos [23].
- A warning from **Ben Tossell**: his Grok bot listed run times for an hourly email-triage job that wasn’t actually running [24]. Check agent-reported status yourself.

Sources

1. X post by @theo
2. X post by @theo
3. So I’ve been testing Gemini 4...
4. Gemini 4 Argon e Muse Wonder
5. Agentic Verification
6. The Most Common Questions About Agent Evals, Answered
7. Context Engineering: How Deep Agents Remember
8. X post by @AnthropicAI
9. X post by @kentcdodds
10. X post by @ClaudeDevs
11. X post by @theo
12. X post by @LangChain
13. X post by @GeoffreyHuntley
14. a new model for open source: closed to pull requests, open to remix.
15. AI@GSB: Ben Tossell, Founder of Ben’s Bites
16. X post by @theo
17. X post by @theo
18. X post by @theo
19. X post by @theo

20. X post by @theo
21. X post by @theo
22. X post by @romainhuet
23. X post by @omarchy
24. X post by @bentossell