A read-only ”explainer” chat beside your coding agent, and the case for lower effort settings

Coding Agents Alpha Tracker

2026-11-01

A read-only “explainer” chat beside your coding agent, and the case for lower effort settings

By Coding Agents Alpha Tracker • November 1, 2026

Salvatore Sanfilippo’s second-chat pattern for keeping up with what your agent built, practitioners moving away from maximum effort on Opus 5.5, Arca’s production harness patterns, and releases from Kody, OpenCode, T3 Code and Hex.

Keep a read-only “explainer” chat next to the agent that’s building

Salvatore Sanfilippo (antirez) describes a problem many agent users run into. After the implementation agent finishes a step, the best next move is a follow-up prompt while its context is fresh. So you skip asking it to explain what it did, and gaps in your own understanding of the project pile up [1].

His fix is a **second chat on the same project, opened read-only**. You tell it that someone else is working in the repo, that it must not write or touch anything, and that its job is to explain architecture and how each part is implemented [1]. You ask the same things every time, so he says this is one of the few cases where a skill file is worth writing. His skill covers: - never edit the project; - if details don’t match, assume the other agent made an edit and recheck; - no jargon; aim to build a mental model in the person asking; - explain specialist details in a short aside; - don’t re-explain topics you say you already know unless asked [1].

This chat can run on a cheaper model. A weaker model may not manage a hard implementation, but it can usually explain one that already exists, for example Sol instead of Astra when you’re near a weekly limit [1]. He doesn’t open it during the first hours of fast iteration. He opens it once long-horizon features start, and says it changed how well he understands details he used to leave to

Opus or Astra [1]. He pairs it with a Claude hook that plays an audio message when Claude is waiting for input [1].

In a separate C lesson he applies the same approach to new code. He works through the low-level details by hand until the architecture is clear, then uses a coding agent to move faster [2].

Effort levels: practitioners are moving off the maximum

After Theo's switch from extra-high to high effort on Opus 5.5, swyx went further: "low effort + critique loop > superultramaxhighwhatever." He frames this as advice for throughput-constrained users and points to his `swyxio/skills/babysit-runs` skill [3]. He also reports that Karpathy used **medium** effort on Opus 5.5 for Hackertube, launched that day. His overall advice: "just tweak to taste" [4].

Arca's production harness: patterns worth borrowing

On Latent Space, Aman Singh described how Arca runs agents on financial work that spans days. Several patterns carry over to coding-agent infrastructure: - **A model-agnostic harness with routing by task tier.** Heavy analysis gets the top tier (Opus-, Sonnet- or heavy-GPT-class models) and background work gets lower tiers [5]. Internal benchmarks drive model choice. Luna looked strong on external benchmarks but was too slow and expensive in their use [5]. - **Shared memory for long-running work.** For each client they store what's happening, who owns what, and what comes next. A nightly "dreaming" process updates it, so a newly started agent can catch up quickly [5]. - **Guardrails at the edges.** Deterministic restrictions decide what agents can do. Fast LLM checks review every database write. Incoming content is screened for prompt injection, and outgoing data gets PII checks plus a consensus of small models looking for exfiltration [5]. Code the agent writes in the sandbox is translated into a strict format and then discarded before it reaches the backend [5]. - **Traces become improvements.** Staff feedback is stored alongside the trace that produced the work. A meta-agent runs over each night's traces and suggests harness or knowledge-base fixes. A human engineer still reviews prompt changes, because agent-written fixes tend to bloat the prompt: "every little fix becomes the whole prompt" [5].

Tool releases

- **Kody v2026.10.31** ships *profiles*. A profile is an isolated identity an agent acts as, with its own grants, expiry and revocation, and profiles can chain into sub-profiles (release) [6].
- **OpenCode** now accepts Claude subscriptions [7]. Theo reads this as Anthropic "loosening the grip on the sub a bit" [8].
- **T3 Code** added `/btw` and side chats, which work with any model or harness [9]. This is a built-in way to ask questions on the side, close

to Sanfilippo's pattern. Theo also shipped a fix (PR #28409) after a user showed the model picker breaking on providers with very long model names and option lists [10, 11].

- **ChatGPT Dot → Codex:** Riley Brown sends Dot an iMessage voice note asking it to start a *local* Codex session, tidy his landing page and push the changes [12]. He says Claude Duet can't yet start local Claude Code sessions. Anthropic's Gabe told him remote control and local Mac access are in progress [12].
- **Hex**, a fast, local voice-dictation tool for talking to agents on Mac, Windows and Linux, has a stable release [13]. DHH added it as a dictation option in Omarchy Dev (PR #17186) [14].

Model takes

- **GPT-6.1 Sol vs GPT-6 Astra:** Will Depue says Sol is "mostly as smart," faster and easier to understand [15]. Theo agrees: it "doesn't demo as well, but it gets to the right answer more often in real work" [16].
- **Google Argon 4:** ThePrimeagen praised it [17]. Theo says it can't work in a real codebase without "egregious" hallucinations and is "10x worse," not 5% worse [18].
- Jerry Tworek says Core Automation has used Codex but is gradually moving to in-house tooling. His view is that Codex is a general tool that "doesn't add that much value on top of just the model" [19].

Short notes

- ThePrimeagen started "agent directed walks" for Omarchy QA. He expects it to be the largest and hardest part of QA to get right, and plans to post long runs early in the week [20].
- Geoffrey Huntley cut overalls.dev builds from 28 minutes to 78 seconds with Buck2/RBE. He is now looking at a ~266 GB closure size [21]. Fast builds matter more when agents rebuild constantly.
- **Security:** Parameter Labs broke out of ChatGPT's sandboxed HTML previews. A message handler in the MathLive library used `innerHTML`, which let them inject an unsandboxed iframe, and a YouTube oEmbed endpoint got around the CSP [22]. A crafted share URL could steal a signed-in user's data or act as them. OpenAI fixed it and paid a $5,000 bounty [22]. If you show agent-generated HTML in a sandboxed iframe, this attack path is worth reviewing.

Sources

1. Un trucco per evitare di non capirci più nulla
2. Impariamo il C, lezione 34: UDP e DNS (parte 1)
3. X post by @swyx

4. X post by @swyx
5. Using Agents for Long-Horizon Financial Work — Aman Singh, Arca
6. X post by @kodykoala
7. X post by @opencode
8. X post by @theo
9. X post by @theo
10. X post by @theo
11. X post by @theo
12. ChatGPT Just Changed A LOT AGAIN (Here's What's New)
13. X post by @opencode
14. X post by @dhh
15. X post by @willdepue
16. X post by @theo
17. X post by @ThePrimeagen
18. X post by @theo
19. Jerry Tworek, Core Automation CEO on AI Reasoning, Agents and Research
20. X post by @ThePrimeagen
21. X post by @GeoffreyHuntley
22. How We Took Over ChatGPT Accounts