Agent-Centered Dev Loops Overtake One-Off Prompting

Coding Agents Alpha Tracker

2026-05-27

Agent-Centered Dev Loops Overtake One-Off Prompting

By Coding Agents Alpha Tracker • May 27, 2026

Top practitioners are reorganizing their dev workflow around agents instead of treating them as sidekicks. Inside: copyable Codex and Claude patterns, DeepSWE's benchmark launch, and the clips/repos most worth studying.

TOP SIGNAL

- **Stop bolting agents onto the old workflow; put them at the center.** Boris Cherny says he ships 20-30 PRs/day by running five Claude agents in parallel, has not written a line of code in over six months, and often leaves hundreds or even thousands of agents running 5-20 hours overnight; he also describes Claude as being at the center of everything inside Anthropic, where the terminal prototype spread quickly across engineering [1]. Romain Huet shows the same operating model in Codex: a plain-English 7am prompt becomes a pinned automation that pulls Slack, Gmail, and calendar context, surfaces top priorities, and drafts emails, while Greg Brockman shared Codex analyzing Slack history and reorganizing channels via computer use into a reusable automation [2, 3].

TRY THIS

- **Build a pinned 7am operating brief in Codex (Romain Huet).** Start by connecting the tools you use every day; Codex has 120+ plugins packaging skills, MCPs, and direct access to tools like Slack, Gmail, and calendar [2]. Then use Huet's pattern: `every morning at 7am, pull all of the context from my Slack, from my Gmail, from my calendar. Give me a full brief of my day, my top five priorities. Point me to what's most critical and urgent ...`

`and draft some of these emails.` Codex turns that into a scheduled daily automation, and you can pin the thread so the same briefing surface is waiting every morning [2].

- **Add a pre-merge review agent before every PR lands (Peter Steinberger).** Steinberger says `autoreview` has been the most impactful skill in his stack after crabbox.sh: it automatically reviews code before landing a PR, finds edge cases, and sometimes runs for hours [4]. If you want a concrete starting point, study the skill spec and port the pattern into your own setup: https://github.com/openclaw/agent-skills/blob/main/skills/autoreview/SKILL.md [4].
- **Tune the operating context before you blame the model.** Theo says GPT-5.5 only clicked for him after about two months, prompting entirely differently, and spending time on `agents.md`; now he says he cannot really use another model for code [5]. Practical routing on top of that: Huet defaults Codex/GPT-5.5 to Medium for most tasks and only uses Extra High for one-off critical refactors, while Matthew Berman's rule is to use frontier models for hardcore upfront planning and cheaper workhorse models for the actual code writing [2, 6].
- **Close the loop on agent fixes.** Palash Shah describes self-optimizing agents as systems that observe their own outputs, evaluate them, and use that signal to improve future performance [7]. The concrete LangSmith Engine recipe: auto-triage feedback on traces, attach an online evaluator to every suggested fix so you do not regress, generate offline evals for your test suite, and keep learning from user preferences over time [7].

WHAT SHIPPED

- **DeepSWE benchmark launched.** Theo calls it the first coding benchmark that actually aligns with how it feels to use these models for coding, and says it exposes much larger divergence than public leaderboards suggest. His read: the gap between official harnesses and simple agent harnesses says a lot, and one comparison showed Gemini 3.5 Flash costing more than GPT-5.5 while scoring about half as high [8, 9, 10, 11].
- **Field test from Salvatore Sanfilippo.** The Redis creator says GPT-5.5 implemented kernels for his MIMO inference system from directions he provided; compared with his handwritten `namasi++` version, the AI-generated implementation ran 2x faster and did not crash past larger context sizes. His current model is mixed: full AI implementation on some projects, line-by-line assist on others, and small manual programs reserved for aesthetics, learning, and new ideas [12].
- **Codex keeps adding practical surface area.** Huet says the app now has 120+ plugins, sandboxed auto review with permission escalation, local or cloud projects, an in-app browser for pointing at UI changes, and a

Codex tab inside the ChatGPT app that launched last week for continuing desktop work from the phone [2]. Theo's recent feature callouts: locked-Mac computer control, a double-Command hotkey to capture the current screen into context, and diff marker settings [13]. Adoption signal: Simon Willison highlighted `@openai/codex` NPM installs growing from about 100,000/day in January to over 1 million/day now [14].

- **Rastermill released for Node agents.** Peter Steinberger extracted image logic into a separate library to stop small hacked images from crashing the process; it is built with Wasm and Rust for speed. Link: http://rastermill.com [15].
- **Undocumented-feature field report.** Theo says he added a new Lakebed feature without announcing it; by morning, Sherlock's agent had discovered it and shipped a curation app on top of it, which Theo says was used perfectly to spec. App: https://badlogic-list.lakebed.app/ with RSS included [16, 17].

GO DEEPER

- **16:12-18:29 — Romain Huet on background computer use.** Best clip today for understanding parallel desktop work: Codex clicks through a simulator, checks for obvious issues, and can create a Reminder without taking over the user's cursor [2].

Wie OpenAI Codex intern wirklich nutzt! Use Cases für Agents (16:12)

- **22:25-24:48 — Theo on verification vs token burn.** Good watch if you are choosing between local-machine verification and cloud swarms: his core argument is that computer-use verification in a real environment can beat simply spinning up more sub-agents and burning tokens [13].

Claude Code vs Codex vs Cursor (an honest comparison) (22:25)

- **27:54-28:29 — Boris Cherny on making Claude the default interface.** Short but important: codebase questions, expenses, and even holiday lookups all route through Claude when the workflow is redesigned around the agent [1].

Claude Code creator Boris Cherny on the end of the software engineer (27:54)

- **Repo to study** — **`autoreview`.** If you only inspect one artifact today, make it Peter Steinberger’s pre-merge review skill spec: https://github.com/openclaw/agent-skills/blob/main/skills/autoreview/SKILL.md [4].

Editorial take: the edge is moving from model shopping to workflow design — persistent context, pre-merge review, and self-evaluating loops beat raw token burn. [2, 4, 7, 13]

Sources

1. Claude Code creator Boris Cherny on the end of the software engineer
2. Wie OpenAI Codex intern wirklich nutzt! Use Cases für Agents
3. X post by @derrickcchoi
4. X post by @steipete
5. X post by @theo
6. Cursor just beat EVERYONE.
7. X post by @palashshah
8. X post by @serenaa_ge
9. X post by @theo
10. X post by @theo
11. X post by @theo

12. Mi chiedo (anche grazie a RIP) perché io sia stato così disponibile alla programmazione automatica
13. Claude Code vs Codex vs Cursor (an honest comparison)
14. X post by @simonw
15. X post by @steipete
16. X post by @theo
17. X post by @thesherlocker