Code Review Loops and Verification Packs Take the Lead

Coding Agents Alpha Tracker

2026-06-11

Code Review Loops and Verification Packs Take the Lead

By Coding Agents Alpha Tracker • June 11, 2026

Today's useful signal was not another one-shot demo. It was the convergence on review-heavy coding-agent workflows: dual-model code review, plan-first execution, PR verification artifacts, and production feedback loops that turn failures into evals—plus fresh releases from Cursor and LangChain, and a new open-source app-builder repo worth dissecting.

TOP SIGNAL

- The strongest pattern today: **code review is becoming the highest-leverage job for coding agents, but only inside explicit loops**. Salvatore Sanfilippo's A-writes/B-reviews/A-revises/B-verifies workflow, Cisco CX's trace → triage → coding-agent → draft-PR pipeline, and Mike Krieger's screenshot/video/staging verification stack all point to the same operating model: let agents read, audit, and self-test aggressively; keep humans on approval and final judgment [1, 2, 3].

TRY THIS

- **Run a two-model review loop.** 1) Let model A write or fix the code. 2) Send the result to model B for review, especially when A stalls. 3) Hand B's review doc back to A for changes. 4) Send the updated code back to B for verification. Salvatore Sanfilippo says this beats vague role splits and is how he turns two models into a macro mixture-of-experts setup [1].
- **Split planning from execution.** Run `/improve` on your strongest model to audit bugs, perf issues, tech debt, missing tests, and future build ideas, then have it write an execution plan that cheaper agents can follow [4]. Mike Krieger front-loads architecture conversations, asks the model to

turn the plan into HTML/markdown/diagrams for team alignment, then routes quick questions to lighter models like Sonnet or lower effort levels when full reasoning is unnecessary [3]. Theo's higher-autonomy variant: give a bounded prompt like `look into other options to make this more performant` and let the model synthesize, test, and validate before reporting back [5].

- **Ship a verification pack with every agent PR.** Krieger's pattern: require screenshots or video on every PR, run real staging flows with real data, cover both known regression paths and the specific intent of the current change, and use video plus FFmpeg when screenshots miss UI jank between frames [3]. When the backend is too messy to boot locally, have the agent build in-memory mocks or proxies so tests can still run and evolve with the codebase [3].
- **Close the production feedback loop like a support queue, not a chat log.** Cisco CX pulls thumbs-down, errors, and low-confidence traces from LangSmith, clusters similar failures, dismisses false positives or opens one Jira per real bug, then hands the case to a coding agent for deeper diagnostics and draft fixes [2]. Humans stay on approval/redirect/final-PR duty, every merged fix becomes a new eval in the repo, and MCP is the swap-any-backend integration layer underneath [2]. This is already running against 10k+ concurrent cases and 153k requests [2].

WHAT SHIPPED

- **Cursor Bugbot update** — over **3x faster**, **22% cheaper**, and finding **10% more bugs**; you can now run `/review` locally before pushing [6]. More: cursor.com/blog/bugbot-updates-june-2026 [7].
- **LangSmith Sandboxes** — now GA; secure, scalable environments for agent code execution, integrated with Deep Agents SDK and LangSmith [8]. More: langchain.com/blog/langsmith-sandboxes-generally-available [8].
- **Managed Deep Agents** — keep the agent definition in your repo, then create and operate managed agents in LangSmith via API [9]. More: langchain.com/blog/introducing-managed-deep-agents [9].
- **RubricMiddleware for Deep Agents** — lets you define what `done` looks like so the agent keeps going until the criteria is met [10]. Deep dive: langchain.com/blog/introducing-rubrics-for-deepagents [11].
- **LangSmith Fleet: Software Engineer template** — Slack-triggered coding agent that takes Linear issues, writes and verifies code, and opens a PR from a sandbox [12]. Try it: langchain.com/templates/software-engineer [13].
- **Open-source project to inspect: Rilable** — Riley Brown says he built the iOS app that generates web and iOS apps with Fable 5 in **10 prompts** for about **$210** in API tokens; each generated app spins up a Daytona sandbox and uses Convex, Vercel AI Gateway, and Chorus iOS skills [14]. Repo: github.com/rbrown101010/rilable [15]. Stack refs: daytona.io [15]

· ios.chorus.com [15] · convex.dev [15] · vercel.com [15].

- **Practitioner comparison worth noting** — Salvatore Sanfilippo says Fable beat GPT 5.5 on a speculative-decoding optimization by reasoning from timing and MoE constraints instead of trial-and-error, but also says it provides fewer intermediate feedbacks and is harder to steer mid-task [16, 1]. Mike Krieger's routing advice lines up with that: use lighter models for quick questions and save higher-effort Fable sessions for work that actually needs them [3].
- **Fable usage reality check** — Theo says usage-based burned **$100 in about 8 minutes**, and he maxed a $200 plan's five-hour session limit in roughly **2 hours** during one workflow; a practical reminder to keep autonomous runs bounded [5].

GO DEEPER

- **4:09–6:17 — Salvatore Sanfilippo on cross-model review.** Best short walkthrough of the A-writes → B-reviews → A-revises → B-verifies loop, and a clean argument against fuzzy `one model designs, one model codes` role splits [1].

Altre considerazioni su Claude Fable (4:08)

- **36:49–40:26 — Mike Krieger on verification loops.** Concrete guidance on requiring screenshot/video artifacts, exercising real staging flows, and using video plus FFmpeg when UI problems only show up between

frames [3].

How Anthropic Uses Claude Fable 5 With Mike Krieger (36:49)

- **Repo worth studying — Rilable.** Worth reading for the architecture alone: Daytona sandbox per app, Convex DB, Vercel AI Gateway, and Chorus iOS skill hooks [15, 14].
- **Template worth skimming — LangChain Software Engineer.** Useful if you want a concrete Slack → Linear → GitHub sandbox flow instead of another abstract agent diagram [12, 13].

Editorial take: the edge is shifting away from raw codegen and toward review infrastructure—clear `done` *criteria, reusable evals, and merge-time verification are starting to matter more than one-shot demos [10, 2, 3].*

Sources

1. Altre considerazioni su Claude Fable
2. Observing And Testing CX Agents | Interrupt 26
3. How Anthropic Uses Claude Fable 5 With Mike Krieger
4. X post by @shadcn
5. Fable is Mythos, and it is really good.
6. X post by @cursor_ai
7. X post by @cursor_ai

8. X post by @LangChain
9. X post by @LangChain
10. X post by @LangChain
11. X post by @LangChain
12. X post by @LangChain
13. X post by @LangChain
14. X post by @rileybrown
15. X post by @rileybrown
16. Claude Fable