SWE-bench Verified is deprecated; WebSockets land in Responses API; AgentMD skepticism goes mainstream

Coding Agents Alpha Tracker

2026-02-24

SWE-bench Verified is deprecated; WebSockets land in Responses API; AgentMD skepticism goes mainstream

By Coding Agents Alpha Tracker • February 24, 2026

SWE-bench Verified is being retired as a frontier coding eval: OpenAI says it's saturated, contaminated, and riddled with test-design issues—SWE-bench Pro is the new recommendation. Also: practical agent workflows (red/green TDD, conformance-test-driven ports), new tool updates (Responses API WebSockets, Codex CLI multi-agent), and a hard look at when AGENTS.md helps vs just adds cost.

TOP SIGNAL

OpenAI is **stopping SWE-bench Verified reporting** and recommending **SWE-bench Pro**, citing **benchmark saturation**, **contamination** (frontier models can regurgitate solutions/problem statements from the Task ID), and **test-design issues** that make a large chunk of remaining tasks effectively unsound to chase [1, 2]. If you're using SWE-bench numbers to pick models or to market agent gains, this is a hard reset on what "good" looks like in coding evals [1, 2].

TOOLS & MODELS

- **OpenAI Responses API — WebSockets mode**
  - New **WebSockets support** aimed at **low-latency, long-running agents with heavy tool calls** (explicitly positioned as good for coding agents) [3, 4].

  - Docs: http://developers.openai.com/api/docs/guides/websocket-mode [3].
  - Huet notes it was built to "keep up" with **GPT-5.3-Codex-Spark** [4].
- **Codex CLI — multi-agent mode**
  - Enable multiple specialized agents in one session (each with its own role/model/behavior) [5, 6].
  - Setup:
    1) Open `~/codex/config.toml` [6]
    2) Add `[features] multi_agent = true` [6]
    3) Run `/experimental` → "Multi-agent mode is now on" [6]
  - Comes with **explorer / worker / general helper** agents out of the box [6].
- **Agentic "full stack orchestration" demo — Antigravity**
  - "Add GPay to your website" via **one prompt**: detects **Angular**, installs deps, edits frontend+backend, then verifies via an automated browser run [7].
- **OpenClaw — new beta**
  - Beta focuses on **security + bugfixes** (and regression fixes), plus adds **Kilo provider** and **Kimi vision + video support** [8].
  - Release notes: https://github.com/openclaw/openclaw/releases [8].
- **Practitioner model notes (Codex vs Claude, cost/latency)**
  - Multiple practitioners are calling **GPT-5.3-Codex + Codex app** the best option "for getting software dev work done," with strong instruction-following (trade-off: more "machine-like" personality) [9]. Brockman attributes this to heavy investment + model/harness co-design + rapid post-training iterations [9, 10].
  - QuinnyPig reports Codex made **Claude Code** feel dramatically weaker after testing (starting from skepticism) [11].
  - Claude Code pain points surfaced today:
    * "Opus 4.6 is thinking WAY TOO long" (annoying, not delivering value) [12].
    * Primeagen tried "Claude fast 4.6" for high-stakes work and spent **$100s in ~1 hour** (but said it was fast) [13].

WORKFLOWS & TRICKS

- **New eval reality: stop optimizing for brittle tests**
  - OpenAI's critique: SWE-bench Verified became less meaningful at high scores—narrow tests can devolve into "guessing" exact names/implementation details rather than measuring coding ability [14].
  - What they say they want next: **longer-term tasks**, **open-ended design decisions**, **code quality/maintainability**, **real-world product building**, and **human-intensive rubric evaluation** [2].
- **Red/green TDD as an agent control surface (Willison)**

  - Prompt pattern: *write tests first → confirm they fail ("red") → implement until they pass ("green")* [15].
  - Why it works with agents: reduces the odds of shipping code that doesn't work or that's unnecessary, and leaves you with a regression suite [15].
  - Copy/paste starter prompt:
    * `Build a Python function to extract headers from a markdown string. Use red/green TDD.` [15]
- **"Conformance suite + reference implementation" makes big agentic ports safer (Ladybird)**
  - Andreas Kling ported **LibJS** to Rust using **Claude Code** and **Codex**, but emphasizes it was **human-directed** (he chose what to port, in what order, and how the Rust should look) [16].
  - Guardrails that mattered:
    * Started with components that had strong **test262** coverage [16].
    * Required **byte-for-byte identical output** vs the C++ pipeline; verified identical ASTs and bytecode; reported **zero regressions** [16].
  - Result: **~25,000 lines of Rust** in **~two weeks** (vs "multiple months" manually) [16].
- **Context files (AGENTS.md / CLAUDE.md): when they help vs when they're just tax**
  - Theo cites a study on "context files" for GitHub issue resolution:
    * Dev-written context files: only **+4%** success vs omitting [17].
    * LLM-generated context files: **-3%** success [17].
    * More exploration/testing/reasoning → **>20% higher costs** [17].
    * Recommendation: **omit LLM-generated context files**; keep only minimal non-discoverable requirements like specific tooling [17].
  - Addy Osmani's rule of thumb: auto-generated AGENTS(.md) duplicates what agents can discover and inflates cost; human-written files help mainly for **non-discoverable gotchas/conventions/landmines** [18]. He suggests treating AGENTS(.md) as a **living list of codebase smells** (not permanent config) [18].
  - Theo's practical heuristics:
    * Don't distract the model with irrelevant background—keep it focused on "the thing" [17].
    * If the info is in the codebase, it often doesn't belong in AgentMD; models can usually find what they need (e.g., via package.json + repo search) [17].
    * If you're investing time, prioritize **unit/integration tests, type checks, and feedback systems** you can expose to the model over growing AgentMD files [17].
- **Agentic quality loops you can steal**
  - **Automated "review → fix → review" loop (Armin**

Ronacher): his `/review` extension for ralph loops between "review on an empty branch" and "go back and fix your shit" until **P0/P1/P2** are resolved [19].
- **Unblock multi-step tasks (Theo)**: if step 2 keeps failing, ask the agent for step 3—he claims it often back-solves step 2 to get there [17].
- **Infra upgrade prompt that actually worked (Ronacher)**: `upgrade me to postgres 18. don't make any mistakes`— shared as a successful approach for painful major version upgrades [20, 21].

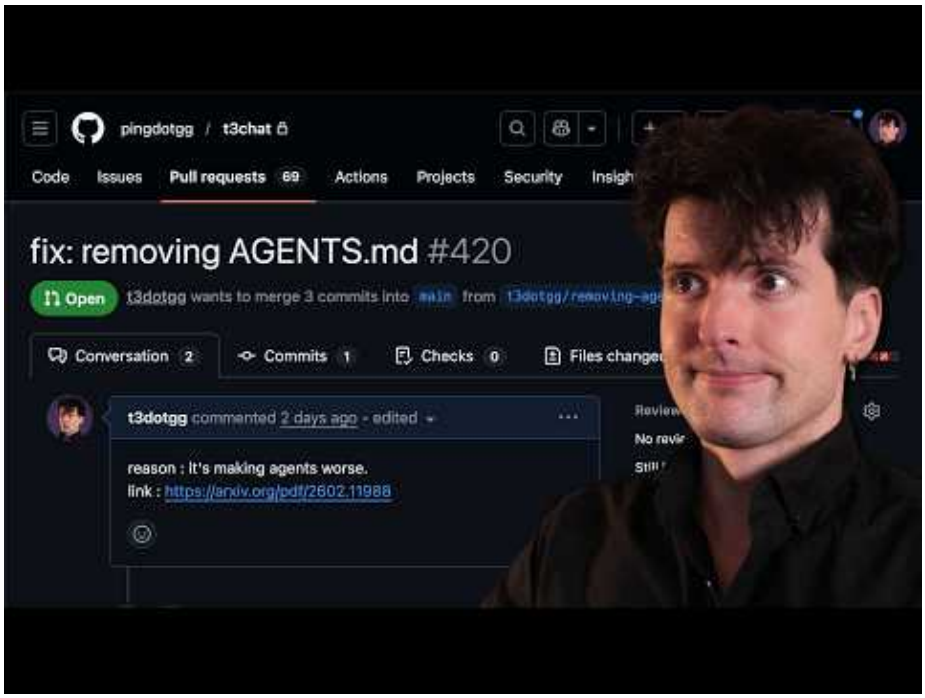
PEOPLE TO WATCH

- **Simon Willison** — launched *Agentic Engineering Patterns* (written by him, not an LLM) and is turning scattered best practices into an evergreen "guide" format [22]. First chapters: "writing code is cheap now" and "red/green TDD" [22].
- **Theo (t3.gg)** — consistently practical on agent context management; argues many AGENTS.md/CLAUDE.md setups are counterproductive and measured as a cost/latency hit [23, 17].
- **Addy Osmani** — sharp framing: AGENTS.md should be about **non-discoverable landmines**, and a single root file won't scale for complex repos (he argues for a hierarchy of scoped files) [18].
- **Kent C. Dodds** — evolving his reviews of agent code toward "is it actually wrong or just different," focusing on principles over personal style; also calls out UI "taste" as a remaining bottleneck (CSS + knowing when UI looks bad) [24, 25, 26].
- **Armin Ronacher** — hands-on, blunt tool feedback: calls MCP architecture token-inefficient/resource-intensive and says it underperforms "skills" in his testing [27, 28].

WATCH & LISTEN

1) Prompt/context hierarchy explained (and why "extra context" sneaks into every request) — Theo (7:10–10:28)

Hook: A concrete mental model for why AgentMD/ClaudeMD "rules" are sticky: provider/system/developer/user layers, and *everything above* gets sent each turn—so context decisions directly impact cost and behavior [17].



Delete your CLAUDE.md (and your AGENT.md too) (7:10)

2) What a "better coding benchmark" should measure — Latent Space + OpenAI Frontier Evals (14:04–15:51)

Hook: The team argues we're moving beyond "solve a small GitHub issue" toward longer-running tasks and harder-to-measure signals like design taste, code quality, and maintainability [14].

SWE-Bench Verified is Contaminated: What Comes Next — with OpenAI Frontier Evals team (14:03)

PROJECTS & REPOS

- **OpenClaw** — beta release notes (security/bugfix focus): https://github.com/openclaw/openclaw/release [8]
- **Agentic Engineering Patterns (Willison)** — guide hub + first chapters:
  - https://simonwillison.net/guides/agentic-engineering-patterns/ [22]
  - https://simonwillison.net/guides/agentic-engineering-patterns/code-is-cheap/ [29]
  - https://simonwillison.net/guides/agentic-engineering-patterns/red-green-tdd/ [30]
- **test262** (referenced as a key "unlock" for safe agentic work on language tooling): https://github.com/tc39/test262 [16]

Editorial take: "Writing code is cheap now," but **proving it's good** (tests, evals, reviews, and anti-contamination discipline) is where serious teams will win [31].

Sources

1. X post by @OpenAIDevs
2. The End of SWE-Bench Verified — Mia Glaese & Olivia Watkins, OpenAI Frontier Evals & Human Data
3. X post by @OpenAIDevs
4. X post by @romainhuet
5. X post by @hqmank
6. X post by @jasonzhou1993
7. X post by @antigravity
8. X post by @steipete
9. X post by @daniel_mac8
10. X post by @gdb
11. X post by @QuinnyPig
12. X post by @jasonzhou1993
13. X post by @ThePrimeagen
14. SWE-Bench Verified is Contaminated: What Comes Next — with OpenAI Frontier Evals team
15. Red/green TDD
16. Ladybird adopts Rust, with help from AI
17. Delete your CLAUDE.md (and your AGENT.md too)
18. X post by @addyosmani
19. X post by @mitsuhiko
20. X post by @mitsuhiko
21. X post by @mitsuhiko
22. Writing about Agentic Engineering Patterns
23. X post by @theo
24. X post by @kentcdodds
25. X post by @kentcdodds
26. X post by @kentcdodds
27. X post by @mitsuhiko
28. X post by @mitsuhiko
29. X post by @simonw
30. X post by @simonw
31. Writing code is cheap now