Structured Agent Workflows Meet Production Reality

Coding Agents Alpha Tracker

2026-06-06

Structured Agent Workflows Meet Production Reality

By Coding Agents Alpha Tracker • June 6, 2026

This brief covers Addy Osmani's SDLC-shaped coding-agent workflow, MongoDB's rollout metrics across ~2,000 engineers, Simon Willison's sandbox build/test pattern, and the most relevant new releases around design-mode UIs, streaming, skills, and deployment.

TOP SIGNAL

Serious coding-agent users are converging on *structured* workflows, not blind autonomy. Addy Osmani's open-source skills system maps directly onto `define → plan → build → verify → review → ship`, with each skill carrying usage guidance, red flags, and verification steps; Simon Willison and swyx describe the same pattern in practice by tightening review, approvals, and testing as stakes go up [1, 2, 3]. MongoDB's rollout makes the stakes concrete: across ~2,000 engineers, coding assistants wrote 70% of last week's checked-in code and engineers produced ~35% more code, but token spend rose enough that the company is explicitly weighing tool cost against future hiring plans [4].

TRY THIS

- **Addy Osmani — run a six-step loop before you let the agent code.** In VS Code/Copilot, start with `/refine` on a vague idea; answer the clarifying questions; run `/spec` to turn that into success criteria and stack assumptions; run `/plan` to generate phased tasks with acceptance criteria and suggested files so the agent stays inside a smaller blast radius; then do `/build`, `/verify`, and `/review` before shipping [1]. Addy's build phase encodes TDD explicitly, and verify/review cover browser testing, security, simplification, CI, git, and docs [1].

- **Addy Osmani + Command Code — keep context thin and route models by phase.** Put reusable behavior in small skill or taste markdown files. Let the agent see only the skill name/description first, then load full content on demand; Addy prefers this progressive loading for context-window economy and reaches for MCP more in auth-heavy cases [1]. Then split models by job: Addy plans with Gemini 3.5 Flash and implements with stronger models like Gemini Pro, Opus, or Codex, while Ahmad Awais says users seed a project's in-repo taste file with Opus or GPT-5.5 and continue iterating with cheaper models [1, 5].
- **Simon Willison — for high-stakes code, split research, build, and red-team passes.** Simon first had GPT-5.5 Pro research MicroPython WASI support and produce `research.md`, then handed that doc to Codex Desktop + GPT-5.5 high to build the package [6, 7]. For the sandbox itself, he watched the trace closely and then used GPT-5.5 to try to break out of it, fixing edge cases it found [2, 7].

  ```
  read the research.md document and build this. You will
  probably need to write a script that compiles a custom
  WASM version of MicroPython as part of this project -
  fetch the MicroPython code to a /tmp directory for this
  as part of that script.
  ```
  [7]
- **swyx — ask for pushback first, then keep autonomy bounded.** A simple prompt trick: frame the task as a question—literally append `?`—so the model critiques the idea and suggests alternatives instead of blindly executing [8]. In his own conference-scheduling agent, he paired that with a bounded while-loop, minimal turns, human approval for proposals, and logs/confirmations across Slack, email, and web [3]. That guardrail matters because, as Kent C. Dodds notes, agents can make you more confident you're on the right path when you're not [9].

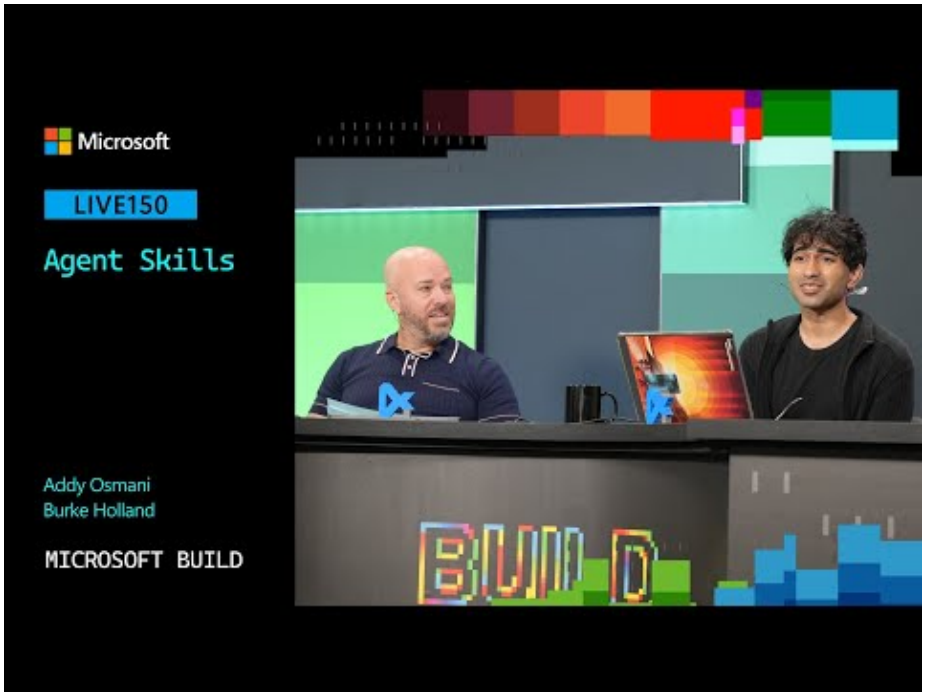
WHAT SHIPPED

- **Simon Willison — `micropython-wasm` alpha + `datasette-agent-micropython` 0.1a0.** Safe Python execution in a MicroPython + WASM sandbox via wasmtime; Simon says he is using it himself after breakout testing with GPT-5.5 xhigh; repos: micropython-wasm, datasette-agent-micropython [7].
- **Cursor — Design Mode.** Point, draw, or talk to update UI; blog: cursor.com/blog/design-mode [10, 11].
- **LangChain — Managed Deep Agents.** Managed, model-agnostic infra for deploying deep agents with one line of code; details: introducing-managed-deep-agents [12].
- **Deep Agents v0.6 — Streaming.** Adds a subscription model for tool and subagent progress in highly parallel systems; examples: streaming-cookbook [13].

- **Skills are productizing fast.** LangSmith Fleet lets domain experts create reusable skills that stay in sync across teams, while Codex now reads `.agents/skills` and plans to deprecate `.codex/skills` [14, 15].
- **Google ADK → LangSmith deployments.** A wrapper turns an ADK runner into a deployment with persistence, streaming, tracing, and built-in endpoints; the demo flow is `wrap(...)` → `uv run langgraph dev` → `uv run langgraph deploy` [16].
- **Command Code (emerging project).** Ahmad Awais says its per-repo `taste` files learn micro-preferences from accepts, edits, and rejects on merge, while its deterministic tool-repair layer has expanded to 16k+ variations across ~600B tokens; one user reportedly ran 12+ hour DeepSeek sessions and 70B tokens through it, and open source is planned very soon [5].

GO DEEPER

- **11:21-15:29 — Addy Osmani: vague idea → clarifying questions → spec.** Watch `/refine` turn a rough GitHub-inspired habit tracker into concrete constraints and candidate directions before any real build work starts [1].



Agent Skills | LIVE150 (11:21)

- **18:16-21:24 — MongoDB: real rollout numbers from ~2,000 engineers.** CJ walks through 70% AI-written check-ins, ~35% more code,

and the cost calculus that follows when token spend starts climbing [4].

Agents in the Enterprise with MongoDB | Interrupt 26 (18:15)

- **7:40-8:16 — Simon Willison: when to trust the agent vs watch it like a hawk.** Short, useful distinction between low-stakes prototyping and security-critical code where you should scrutinize every step and then run adversarial tests [2].

Scott and Mark learn to Vibe Check with Simon Willison | LIVE113 (7:40)

- **Repo to study — micropython-wasm, datasette-agent-micropython, and Simon's research.md.** Good reference for research-model-first, coding-agent-second workflows plus WASM sandbox design with host functions and fuel limits [7, 6].
- **Repo to study — streaming-cookbook.** If you're building multi-tool or multi-subagent systems, this is the fastest way to see what useful streaming telemetry should look like in practice [13].

Editorial take: the biggest edge right now is not 'more autonomous agents'; it's tighter scaffolding around them—phases, slim context, approvals, and hostile verification.

Sources

1. Agent Skills | LIVE150
2. Scott and Mark learn to Vibe Check with Simon Willison | LIVE113
3. Scott and Mark learn to Vibe Check with Swyx | LIVE115
4. Agents in the Enterprise with MongoDB | Interrupt 26
5. Making DeepSeek v4 outperform Opus 4.7 with Taste — @AhmadAwais , CommandCode.ai
6. Running Python code in a sandbox with MicroPython and WASM
7. Running Python code in a sandbox with MicroPython and WASM

8. X post by @swyx
9. X post by @kentcdodds
10. X post by @cursor_ai
11. X post by @cursor_ai
12. X post by @LangChain
13. X post by @LangChain
14. X post by @LangChain
15. X post by @embirico
16. Deploy Google ADK Agents on LangSmith Deployment