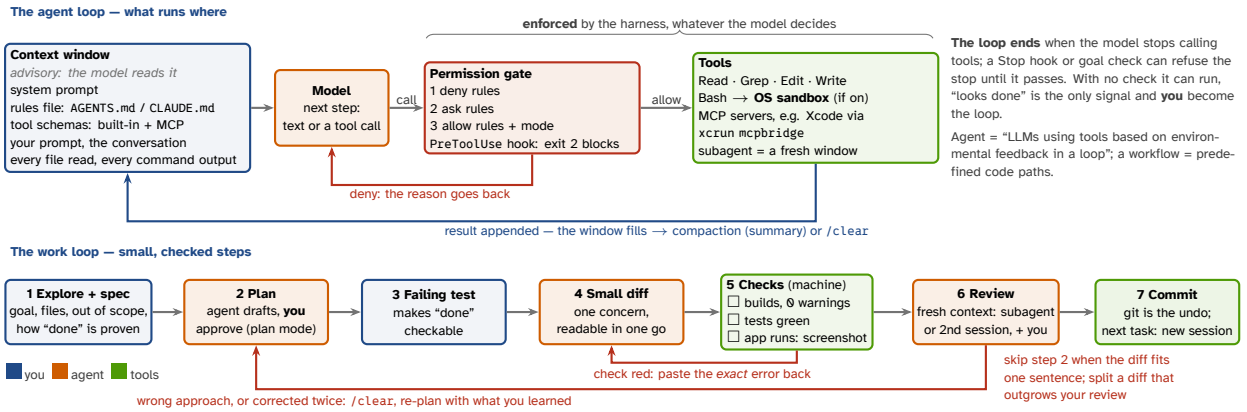
AI coding agents at work — loop, context, permissions, proof

In one line: A coding agent is a model calling tools **in a loop** inside a **harness** that decides what each call may touch. You control the **context** it sees, the **permissions and sandbox** around it and the **check** it must pass: rules files advise; permission rules, hooks and sandboxes enforce; a test, build or screenshot proves.



Rules files — who reads what

file	read by · how it is found
`AGENTS.md`	open format; Agentic AI Foundation (Linux Foundation) with MCP and goose since 9 Dec 2025; read by 20+ tools. **Nearest file wins**; the chat prompt overrides.
Codex	`~/.codex`, then repo root → cwd; `AGENTS.override.md` first; concatenated, closer = later; stops at 32 KiB.
`CLAUDE.md`	Claude Code: managed → user → project → `CLAUDE.local.md`; ancestors at launch, subdirs on demand; **concatenated, none overrides**; aim <200 lines; `.claude/rules/` by `paths:`; `AGENTS.md` only with no `CLAUDE.md` — and a private `CLAUDE.local.md` counts.
Copilot	`.github/copilot-instructions.md`, `*.instructions.md` (`applyTo`); agent, review, CLI: nearest `AGENTS.md` or root `CLAUDE.md/GEMINI.md`.

- **Context, not configuration**: "never push" is a request — enforce it with a deny rule, a hook or the sandbox. A rule it keeps ignoring: the file is too long.

```
# AGENTS.md (Claude Code reads it when there is no CLAUDE.md)
Build: xcodebuild -scheme App build      # warnings = errors
Test:  xcodebuild test -scheme App -only-testing:AppTests
Stack: Swift 6 language mode; iOS 18+; SwiftUI
Rules: @MainActor view models; no force unwraps; no new deps
Done:  clean build, tests green, screenshot of the screen
Never: edit .pbxproj by hand, read Secrets.xcconfig, push
```

Context is the budget

- The window holds system prompt, rules, tool schemas and *every* message, file read and command output; quality degrades as it fills. Chroma's "context rot" (Jul 2025, 18 models): accuracy falls with input length even on simple tasks.
- **One task per session**, `/clear` between; after two failed corrections, clear and re-prompt with what you learned.
- Keep exploration out: a **subagent** reads in its own window and returns a summary; skills and path-scoped rules load on demand. APIs newer than the training data (iOS 27 SDK, Swift 6.4): hand it the docs page.

Permissions & sandboxes

Claude Code mode	runs without asking
`default` (Manual)	reads only
`acceptEdits`	+ edits, `mkdir touch mv cp` in the working dirs
`plan`	reads; no source edits until the plan is approved
`auto`	everything; a classifier model reviews risky actions
`dontAsk`	only pre-approved tools, the rest denied — CI, scripts
`bypassPermissions`	everything — isolated containers / VMs only

- Rules: **deny** → **ask** → **allow, first match wins**, specificity ignored: `Bash(aws *)` denied beats `Bash(aws s3 ls)` allowed; deny holds in every mode. Enforced by the client, not the model.
- **Sandbox** (shell only, off by default; Seatbelt, bubblewrap): writes = working dir + temp; reads include `~/.ssh`, `~/.aws` until denied; network via a proxy, allow-list starts empty. File tools, MCP servers, hooks run **outside** it. Fails **open** unless `failIfUnavailable`.
- A rule matches *text*: deny `Bash(git push *)` misses `git -C . push`, a `Read` deny misses a script that opens the file. Limits that ignore the text: the sandbox.

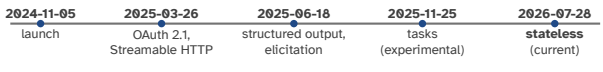
Failure modes → the check

- **Stale or invented API**, worst past the cutoff — compile, read the docs.
- **Silenced symptom**: `@unchecked Sendable`, `nonisolated(unsafe)`, `try!`, a weakened test — read the *test* diff first.
- **Plausible, unverified**: demand evidence (test output, command + result, screenshot). "If you can't verify it, don't ship it."
- **Lost undo**: checkpoints miss changes made through Bash — commit often.
- ! A "Never" line in `CLAUDE.md` as a security control; `bypassPermissions` outside a VM; an unread MCP server — full access, outside the sandbox.

Xcode's agents, release by release

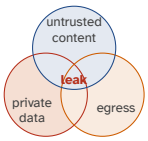
- **26** (2025): ChatGPT and Claude accounts, any Chat Completions provider by API key, or a **local model** on Apple silicon; Coding Tools; predictive completion runs locally.
- **26.3** (RC 3 Feb 2026): **agentic coding** — Claude Agent + Codex; a permissions system; Xcode's tools over **MCP** to any agent, project open: `claude mcp add --transport stdio xcode -- xcrun mcpbridge`.
- **27** (14 Sep 2026; Swift 6.4; Apple silicon only): **plan** = editable Markdown, approved before it proceeds; + Gemini, Antigravity; agents boot simulators, tap and screenshot, drive the debugger, schemes, build settings, entitlements over MCP; **plug-ins** = skills + MCP servers + Agent Client Protocol agents; `lldb-mcp`; an opt-in filesystem security layer over agents and every process they spawn.

MCP — how tools plug in



- JSON-RPC 2.0: **host** → a **client** per server → **server**: **tools**, **resources**, **prompts**; stdio or Streamable HTTP; dated by the last breaking change.
- **2026-07-28**: no `initialize`, no sessions — each request carries version + capabilities; tasks, MCP Apps = extensions; Roots, Sampling, Logging deprecated.
- Tools are **arbitrary code execution**; their descriptions are untrusted unless the server is; consent before every call.

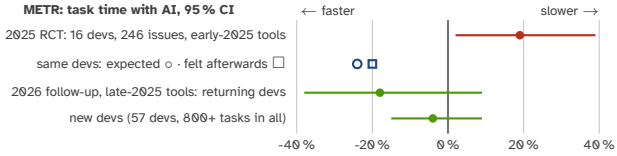
Security — the agent is an attack surface



Lethal trifecta (Willison, Jun 2025): an agent with all three can be told, by text it reads, to send your data out. Remove a leg: no secrets in reach, no untrusted input, or no egress. Filters that stop "95 %" are "a failing grade".

- **Prompt injection** (OWASP LLM01:2025): orders hidden in a page, issue, README, dependency or tool result; RAG and fine-tuning do not fully mitigate it.
- **Hallucinated packages** (USENIX Sec. 2025; 16 models, 576 000 samples): ≥5.2 % (commercial) / 21.7 % (open) of package names do not exist — 205 474 an attacker can register. Check and pin each new dependency.

What the measurements say



- 2025: **+19 %** time, felt 20 % faster; the 2026 follow-up leans faster, but METR calls it unreliable (developers declined no-AI tasks). **Felt ≠ measured.**
- **SWE-bench** (ICLR 2024): 2 294 GitHub issues, 12 Python repos; solved = hidden tests pass; Claude 2: 1.96 %. **Verified**: 500 human-checked. OpenAI dropped it (23 Feb 2026): of 138 problems o3 did not reliably solve, 59.4 % had flawed tests or descriptions (35.5 %: tests that demand one implementation); every model tested recalled gold patches.

Remember: context in, permissions around, checks out — the agent types, you sign.

Related: `llm-features-in-apps` · `xcodebuild-simctl-cli` · `ci-pipeline-design` · Supply-chain security · Threat modelling · XCTest & Swift Testing · Git in depth