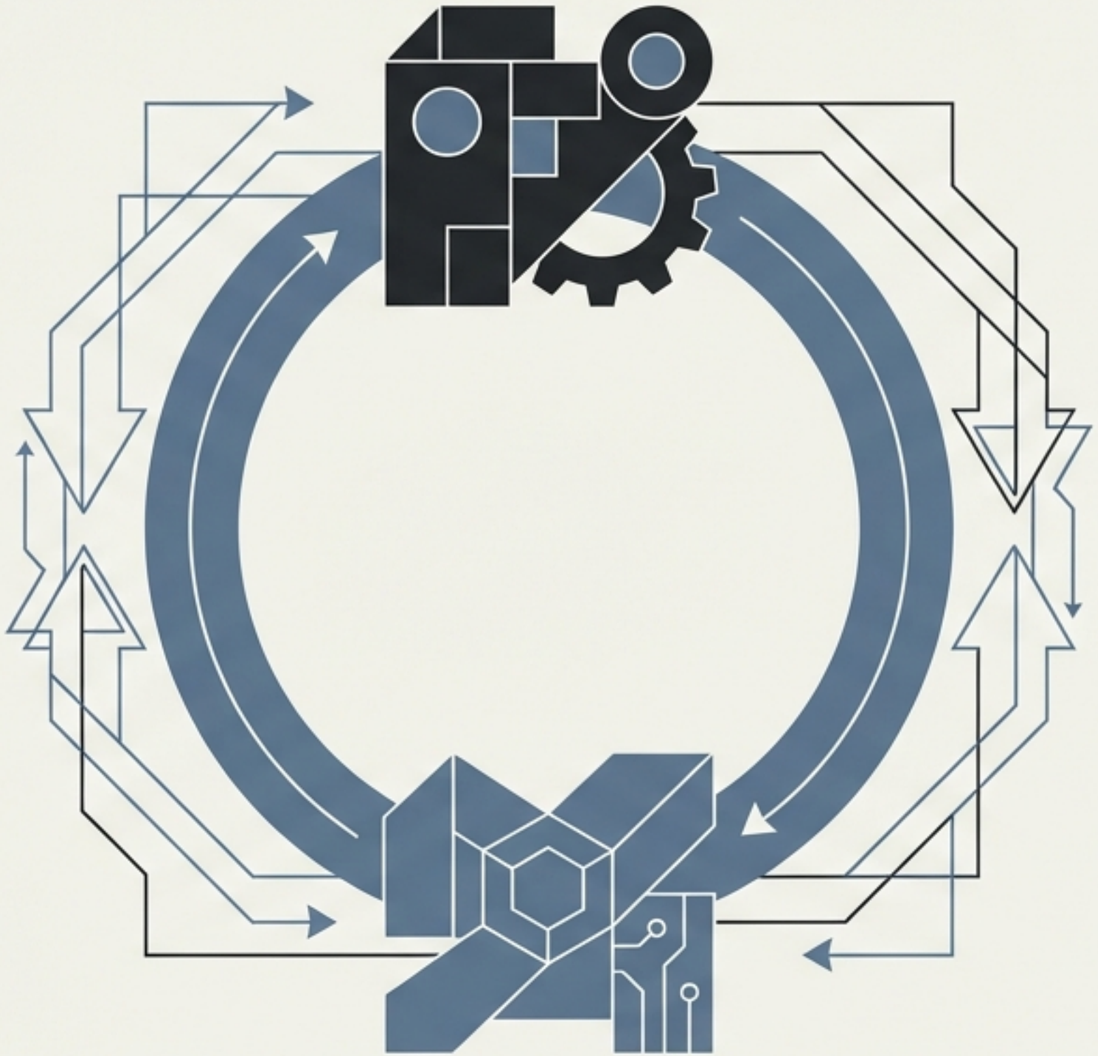
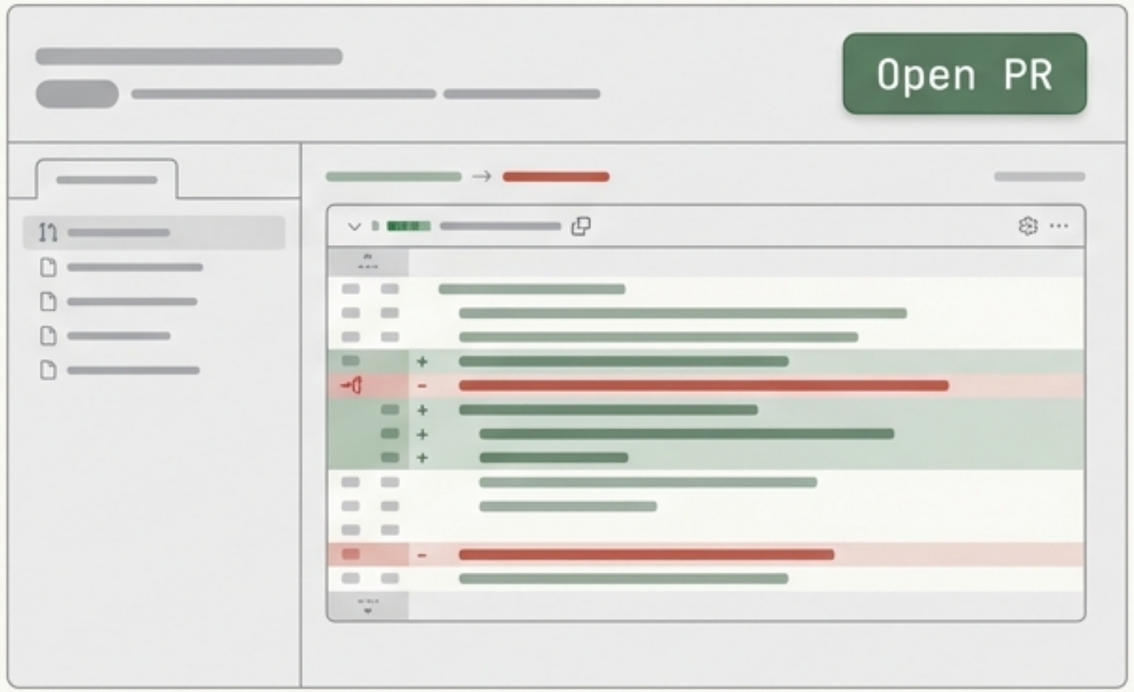
Recursive Refinement



A routine pull request to optimize agent efficiency.



- Submit a pull request to `CrewAI` adding a KCP manifest and TL;DR summary files.
- Share the benchmark efficiency result.
- See if the maintainers want to merge it.

76%

reduction in agent tool calls.

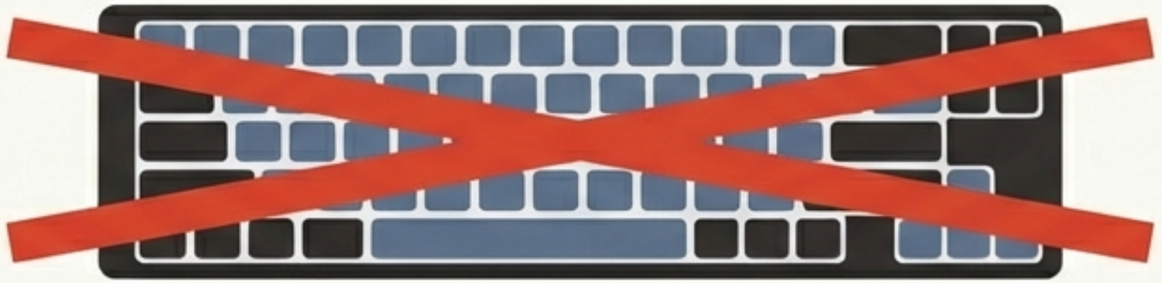
The first reviewer was not human.

Cursor Bugbot

The Automated Reviewer

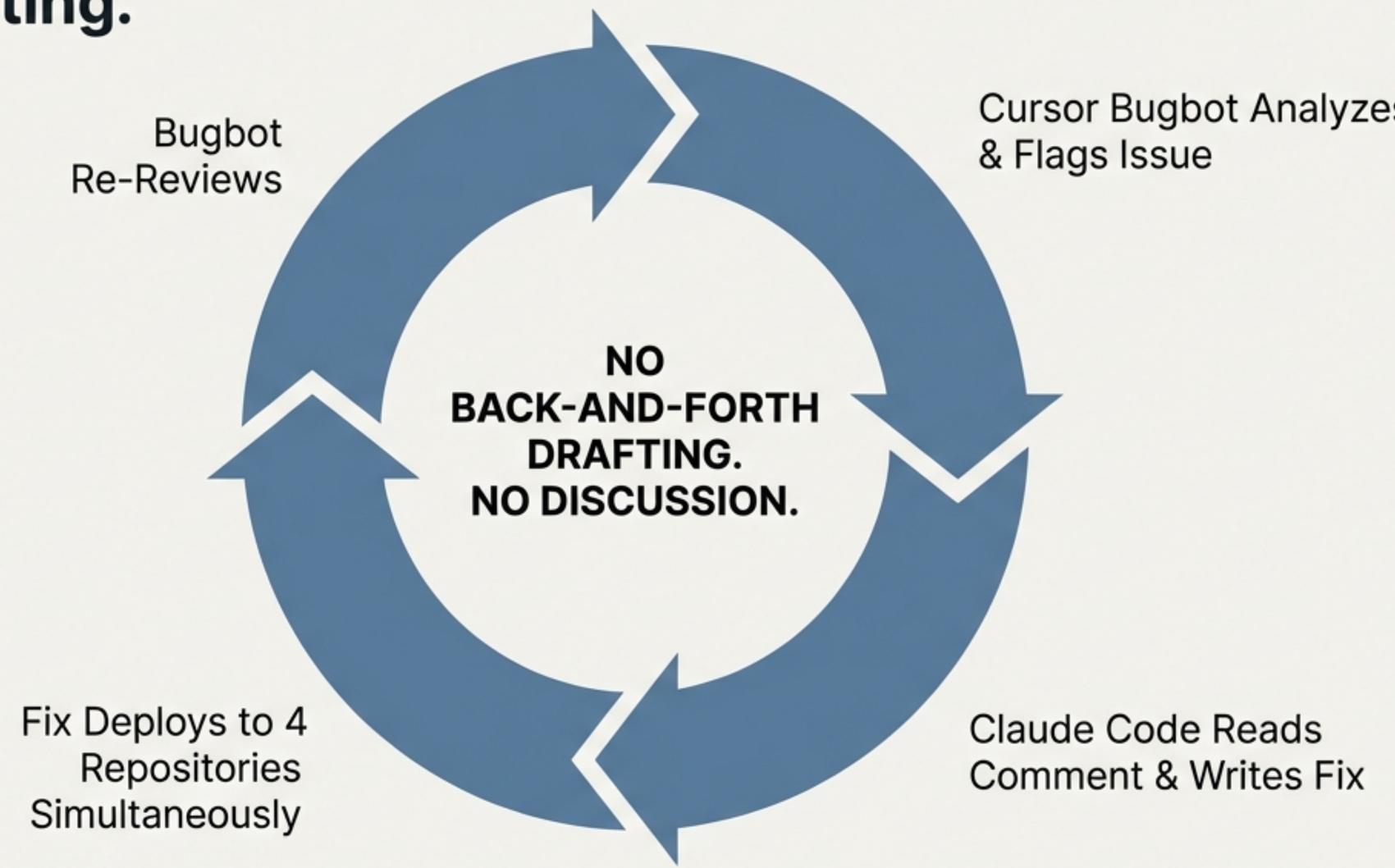
Claude Code

The Automated Fixer



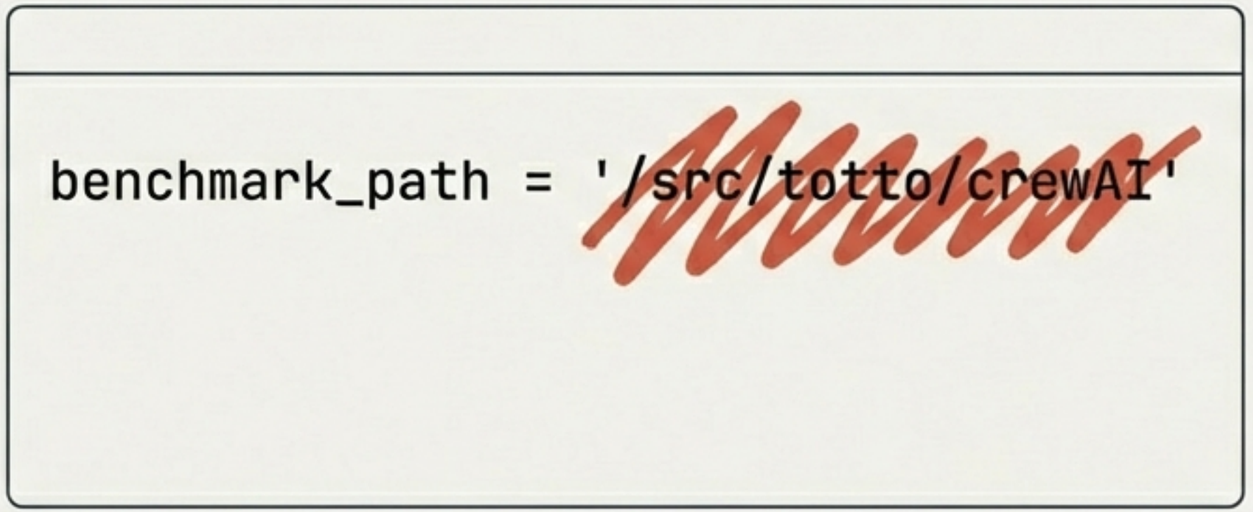
Cursor Bugbot, an automated review system running on incoming `CrewAI` PRs, found an issue. Claude Code fixed it. It happened again. Six times in a few hours.

The autonomous review loop operates entirely without human drafting.



Round 1 catches an embarrassing local hardcoded path.

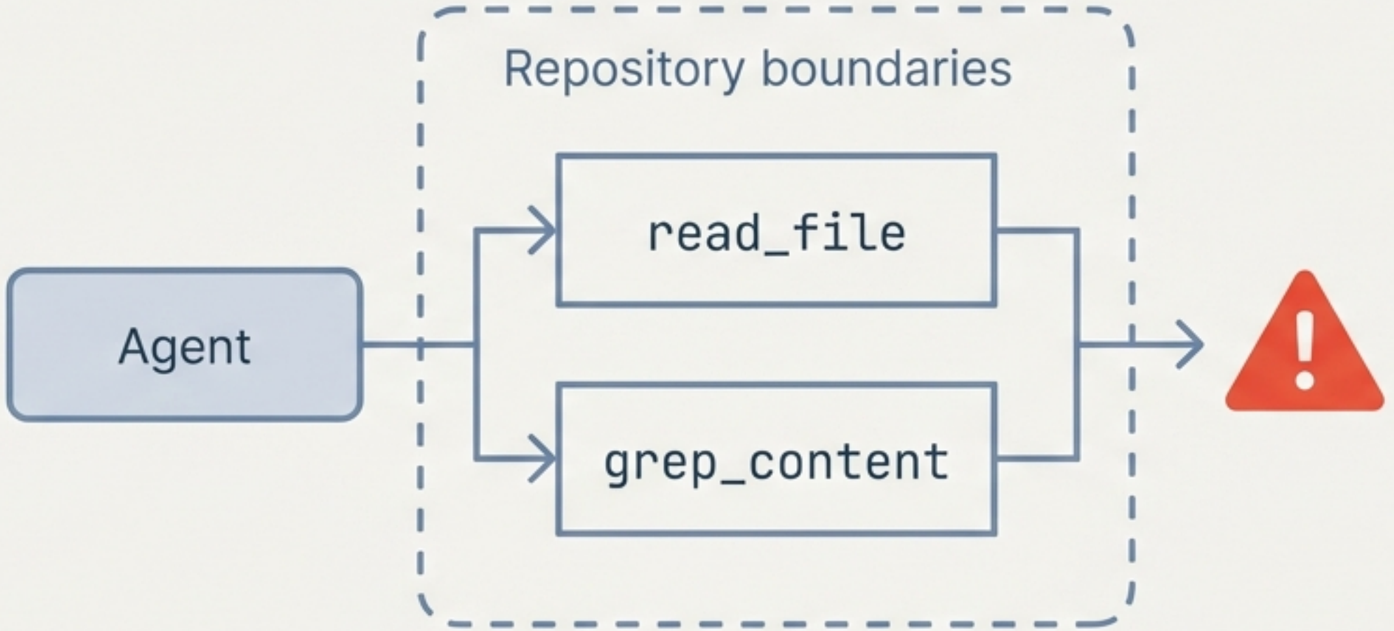
Layer 1: Surface



A copy-paste mistake in the benchmark script. The path works locally but breaks immediately on any other machine. An obvious, fair catch.

Round 2 flags an unrestricted file reading vulnerability.

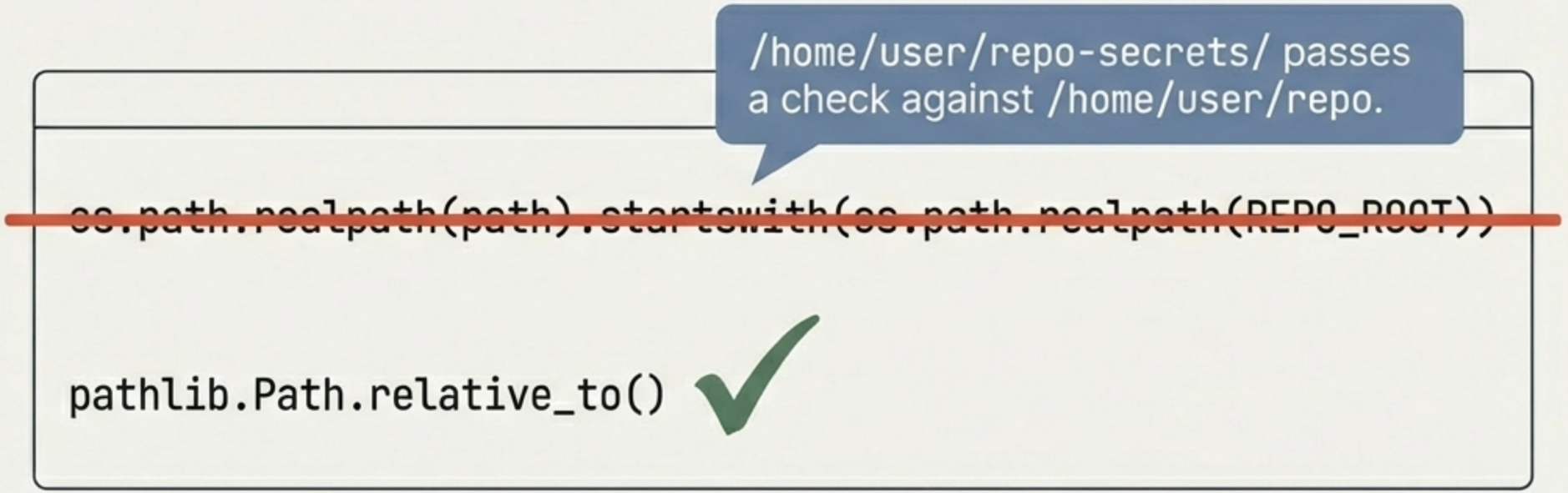
Layer 2:
Functional Safety



The benchmark runner tools lacked path restrictions. The bugbot caught the exposure, prompting the addition of explicit path validation.

Round 3 exposes a subtle path traversal exploit in the validation logic.

Layer 3:
Security Logic

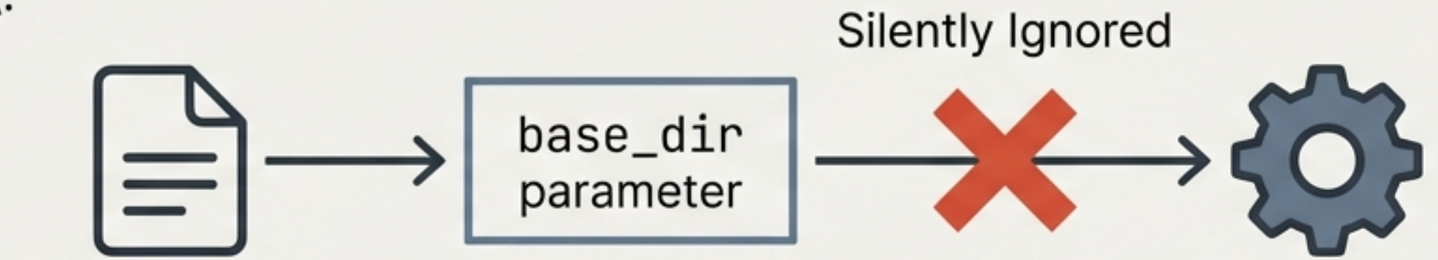


The bugbot correctly identified the traversal vulnerability and also noticed the `glob_files` tool was completely unprotected.

Round 4 detects API contract violations and argument injection vectors.

Layer 4:
Execution &
API Contracts

Bug A:

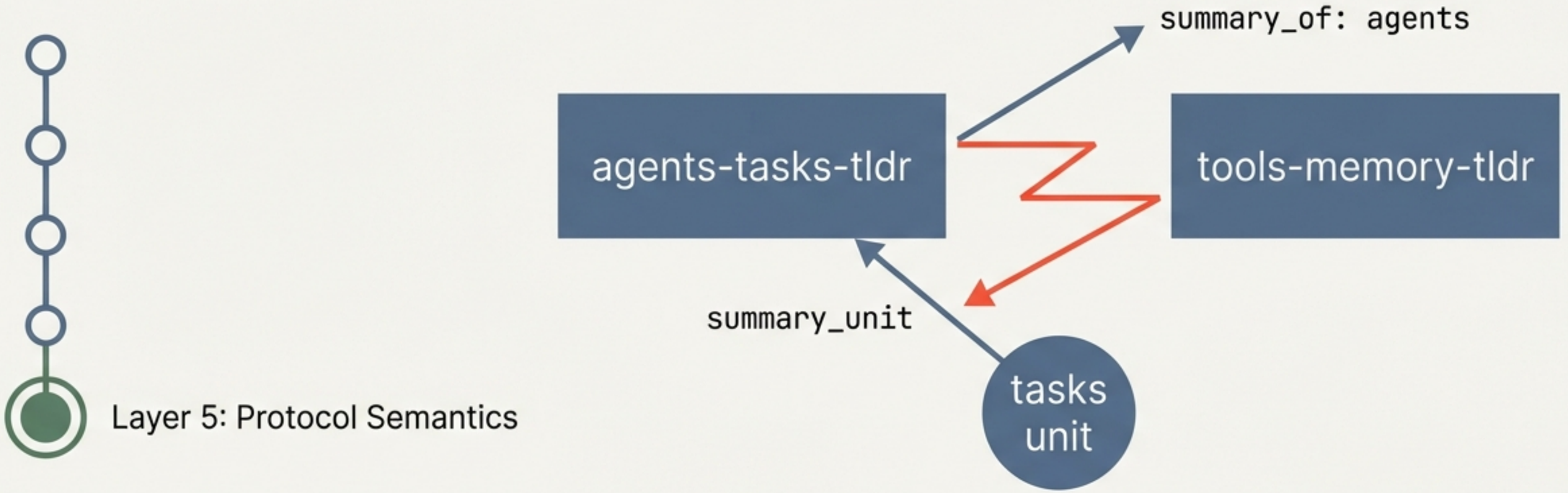


Bug B:

```
subprocess.run(['grep', pattern, filepath])
```

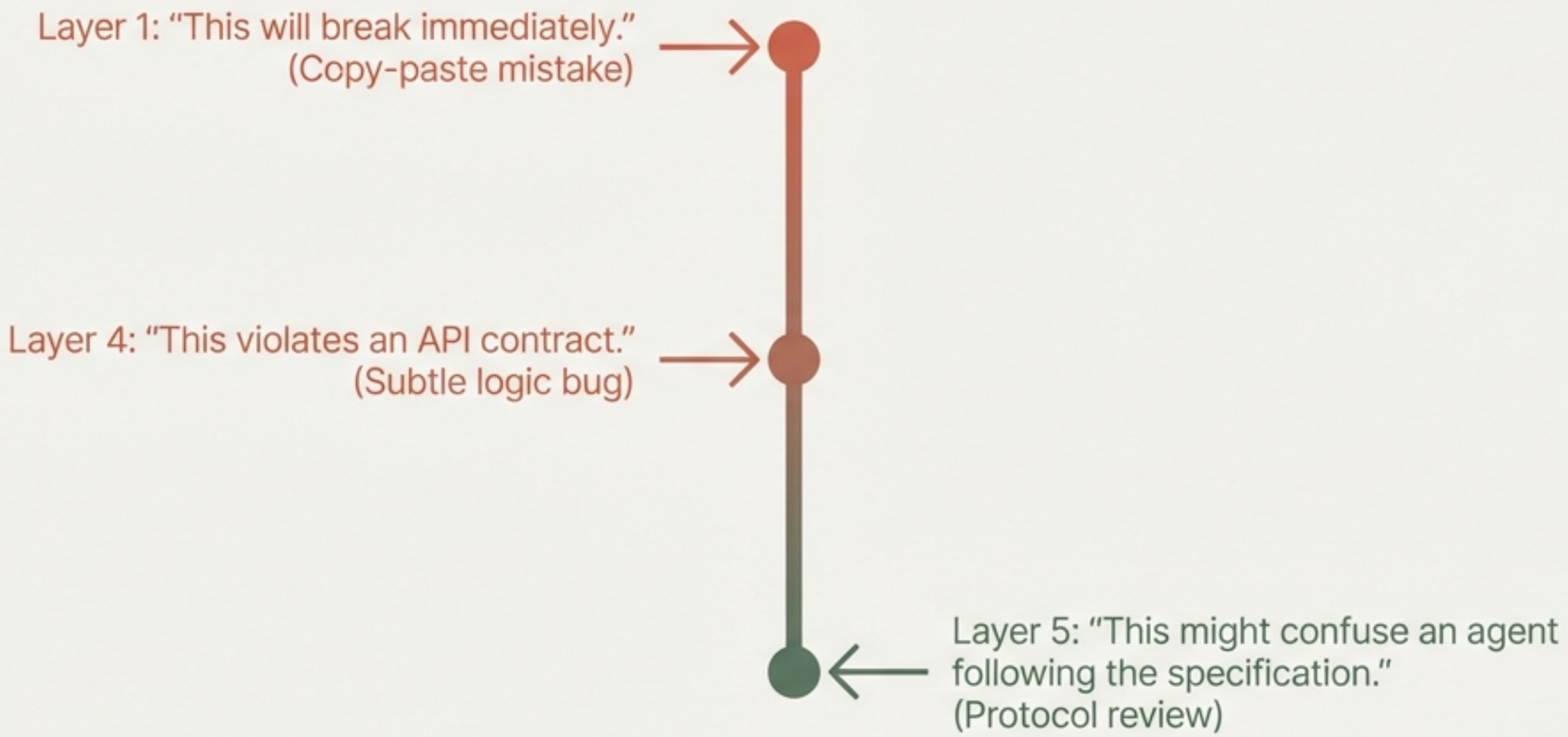
Missing –e flag. Patterns starting with a dash become malicious arguments.

Round 5 escalates from linting to a protocol design review.



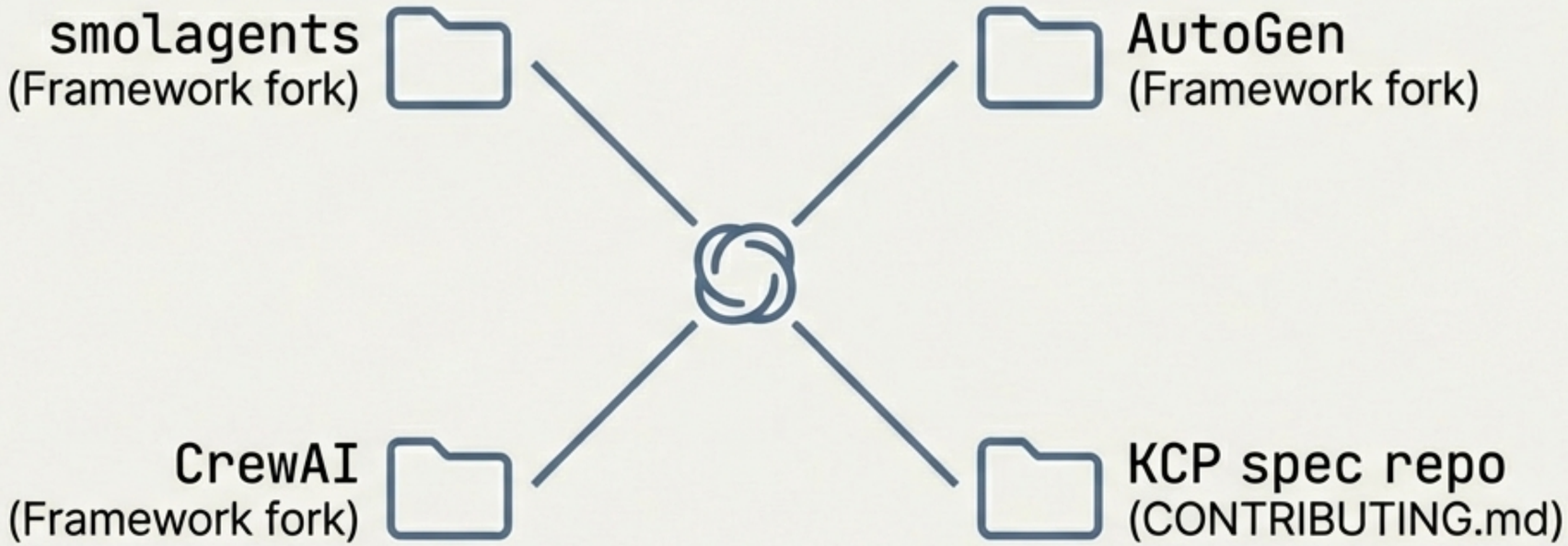
The bugbot realized that a KCP-aware agent might skip the TL;DR shortcut entirely if the semantic back-pointers were misaligned. The fix required restructuring explicit context relationships to remain spec-compliant.

Each fix closed an obvious gap to expose a subtler one beneath it.



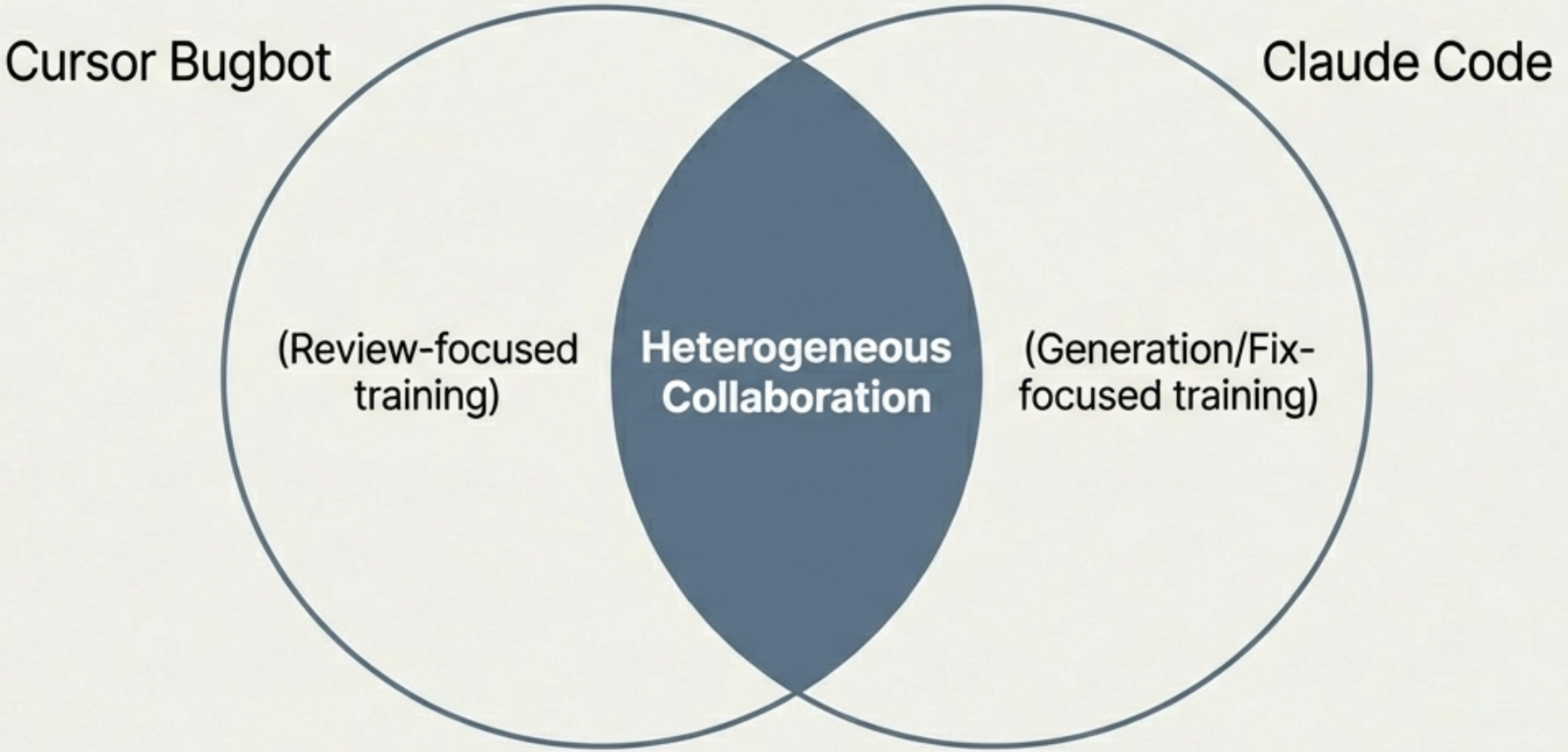
The progression over six iterations was not a coincidence. The automated system systematically worked through layers of complexity.

Fixes synchronized across four repositories simultaneously.



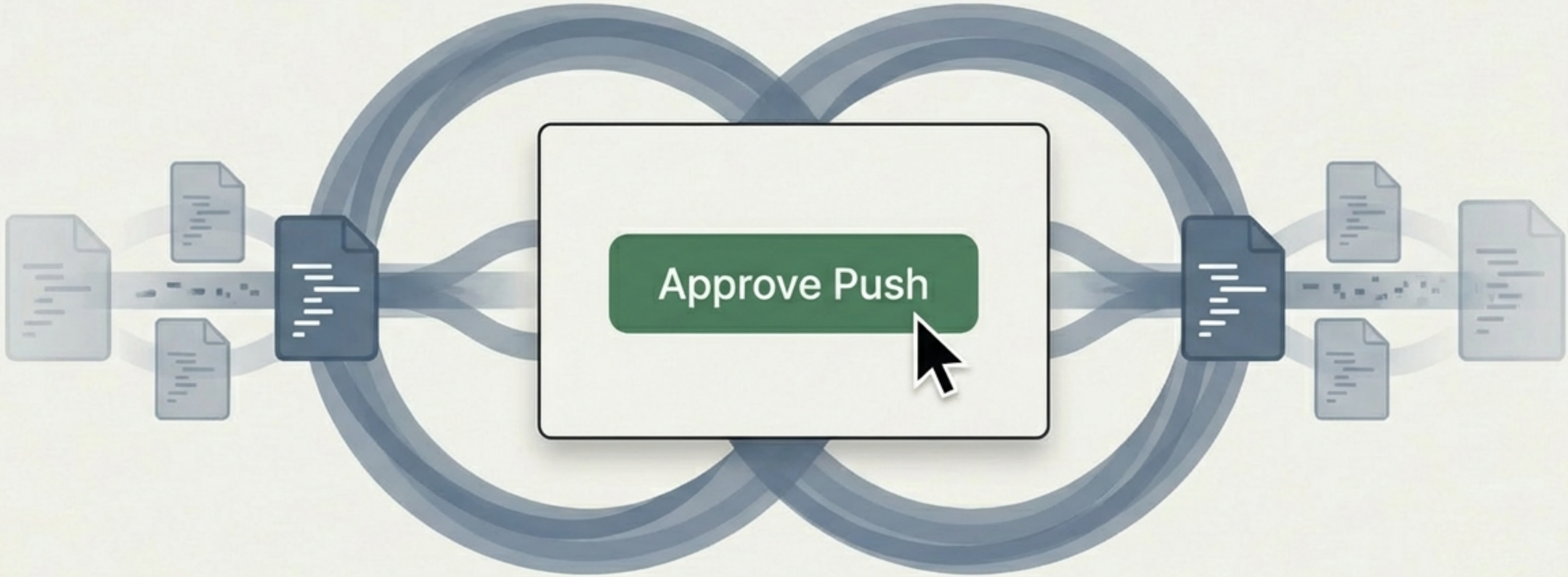
Because the benchmark runner pattern is shared, every round of fixes was coordinated and deployed across all four locations without any human direction.

Heterogeneous AI models catch what single models miss.



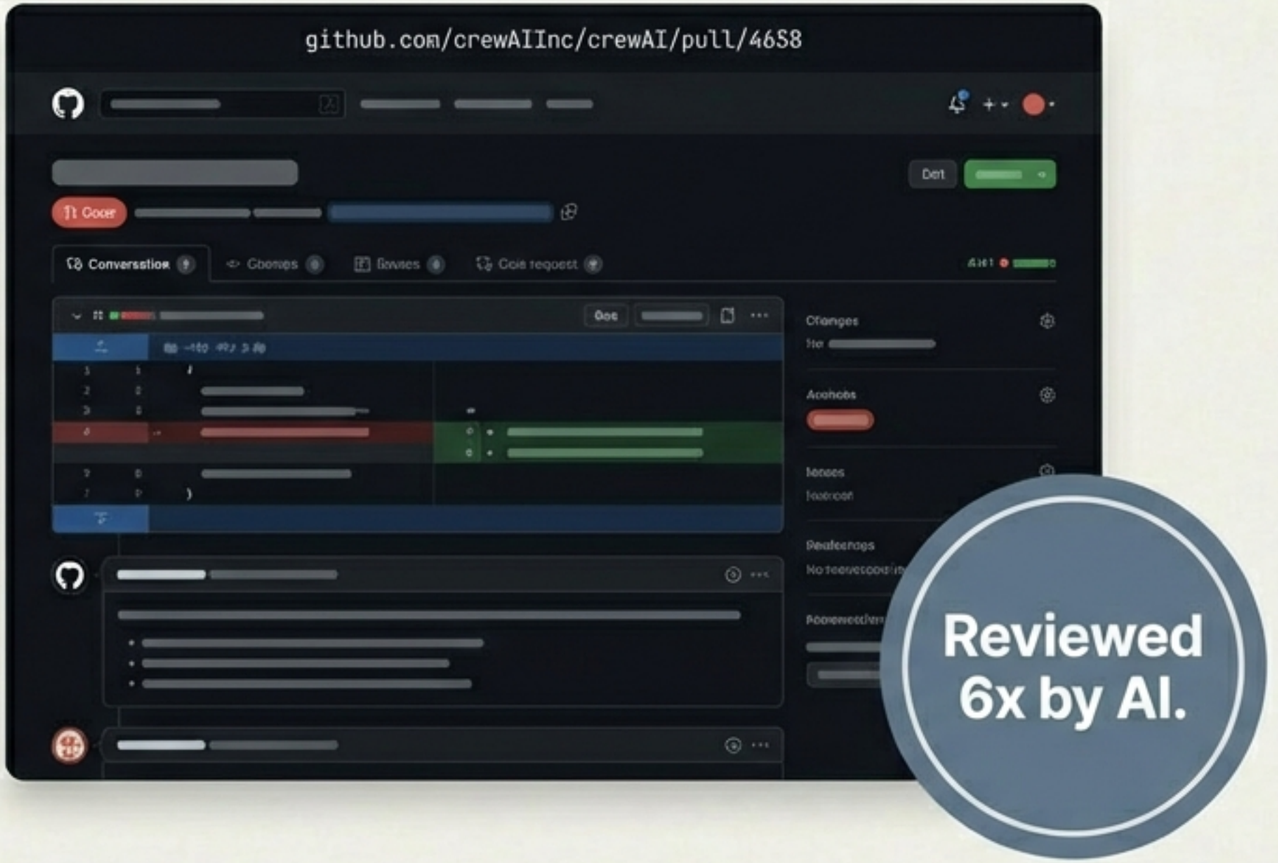
Cursor Bugbot and Claude Code have different training, different strengths, and different blind spots. Combined, they caught subtleties a human would have confidently shipped.

The developer transitions from code writer to system supervisor.



The human role consisted entirely of watching the process happen and approving the pushes. A sequence of increasingly specific issues resolved without manual drafting.

The code is considerably tighter before a human ever opens the pull request.



The PR is still open. No human reviewer has looked at it yet. But by the time they do, the architecture, security, and syntax have already survived rigorous recursive refinement.