AI Reviews AI

Closing the Loop in Agentic Development

think tecture

Daniel Sogl
@sogldaniel
Consultant @ Thinktecture

About me

Daniel Sogl

- Consultant @ **Thinktecture AG**
- MVP — Developer & Web Technologies
- Focus: Developer Productivity & Generative AI
- Socials: **linktr.ee/daniel_sogl**

AI writes the code now

~42% of new code is AI-assisted

Source: Sonar — State of Code Developer Survey (1,100+ devs, self-reported)

So we ship a lot more

GitHub Octoverse 2025

+29% merged pull requests year-over-year

More code. More features. More pull requests. Faster than ever.

Review is the new bottleneck

We ship faster than anyone can review it

think tecture

The Productivity–Reliability Paradox

Faros AI — "Acceleration Whiplash", 22,000 developers (2026)

+210%

tasks involving code

+441%

time in PR review

+31%

more PRs merged *unreviewed*

+243%

incidents per PR

Individual output soared. Bugs, incidents, and skipped reviews came with it.

Vendor telemetry — read as directional, not peer-reviewed.

The problem: plausible code

Code that looks correct is the hardest to review

- Compiles ✅ passes existing tests ✅ looks right ✅
- …and still hides a latent flaw at some edge case
- Machine-specific error profile: invented APIs, hardcoded values, subtle security holes

Veracode (100+ LLMs, 80 tasks): **45%** ship an OWASP Top-10 flaw — and the Spring 2026 re-test still sits at **~55%** pass rate. Syntax got better; security didn't.

Quality Gates

Make correctness something you can check automatically

Two kinds of sensors

Martin Fowler — "harness engineering"

Deterministic

Tests · linters · type checks · hooks

Fast · cheap · same answer every time

Probabilistic

AI review · "LLM as judge"

Context-aware · semantic · *non-deterministic*

Build on the deterministic ones. Add the probabilistic ones on top.

The layered defense

Reliable checks first, AI judgement on top

Layer 1a — Agent hooks

Checks the agent can't skip

```
// .claude/settings.json → "hooks": { ... }
"PostToolUse": [{
  "matcher": "Edit|Write", // after each file edit — fast, scoped
  "hooks": [{ "type": "command",
    "command": "eslint --fix \"$(jq -r .tool_input.file_path)\"" }]
}],
"Stop": [{ // the hard gate: before the agent may finish
  "hooks": [{ "type": "command", "command": "tsc --noEmit && vitest run" }]
}]
```

- The edited path comes in on **stdin** (`.tool_input.file_path`) — check just that file
- PostToolUse feeds the error **back**; the agent reads it and self-corrects
- `Stop` is where you *block*: full typecheck + tests must pass to end the turn

Layer 1b — Tests are the spec

TDD / BDD: you own the spec, the AI owns the implementation

- Write (or review) the failing test **first** → commit it
- Agent writes code until it goes green — it can't validate its own bug
- Guard against tampering: `git diff tests/` · rule: *never edit tests to pass*
- This is the only layer that checks **business correctness**

Layer 2 — Review agent in the loop

Probabilistic, but local & fast

- A reviewer sub-agent critiques the diff *before* the PR exists
- Catches smells, missing edge cases, unclear intent
- Runs in seconds, on your machine, in context
- Watch the noise: too many false flags and you learn to dismiss all of them — real bugs included

> Run it every iteration for early signal, and let the deterministic gates decide.

Layer 3 — At the PR

Two archetypes of cloud review

GitHub Copilot

The zero-setup default

native to GitHub, already in your Copilot seat

- Reviews in context: reads related files, traces imports
- Conservative: fewer comments, fewer false positives
- Generalist, light on deep security
- GitHub & Azure DevOps

Greptile

Whole-codebase depth

indexes the full repo into a semantic graph

- Catches cross-file bugs diff-only tools miss
- Thorough: depth over speed
- Slower, noisier, pricier
- GitHub / GitLab / Bitbucket

Who reviews the reviewer?

Don’t let AI rubber-stamp its own AI code

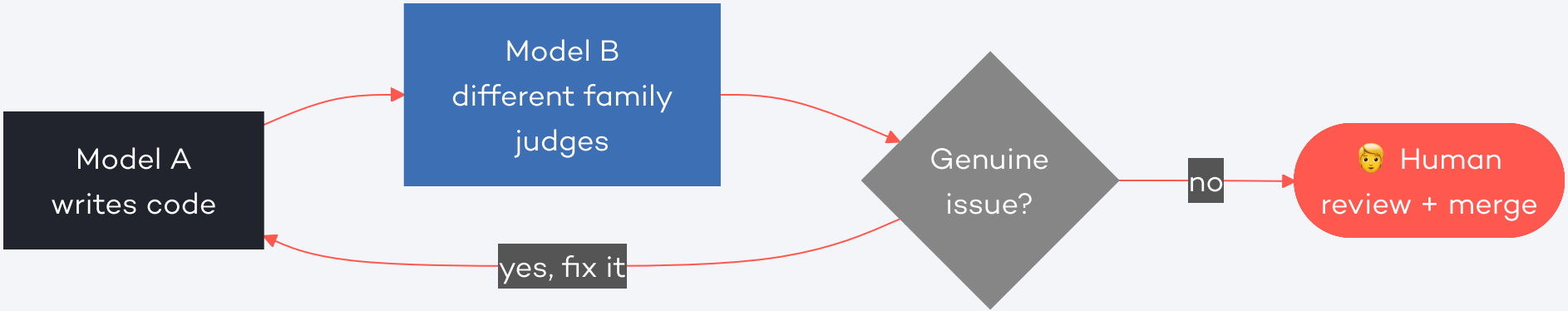
Same model → same blind spots

- AI writes *plausible* code — and rates *plausible* code highly
- The bias is **family-level**: a GPT judge favors GPT, a Claude judge favors Claude
- 2026 frontier judges over-rate their own family by ~10–25% — up to ~80% head-to-head
- …but it's noisy: some Claude models even *under*-rate their own. Don't over-claim.

So let a different model judge the code.

Rubber ducking across models

Model A writes, a *different-family* Model B judges



Cross-family judging breaks the self-preference loop. Panels and order-randomization help further.

Takeaways

A self-correcting loop you still control

- Review is the new bottleneck; generation is the cheap part now
- Start with the **deterministic** gates: hooks, linters, tests
- Tests carry **intent**: write the spec, let the agent fill it in
- Keep AI review a supplement; let a **different** model do the judging
- Keep a human at the start and the end of the loop

Thank you!

Questions?

think tecture

Slides & socials

linktr.ee/daniel_sogl

thinktecture.com

daniel.sogl@thinktecture.com