The Multi-Agent Review Mindset

Code review isn't a flat checklist; it's a hierarchy of needs ranging from syntax to system design. Every reviewer brings a different lens to the code. Understanding these mindsets helps you build a more complete review process.

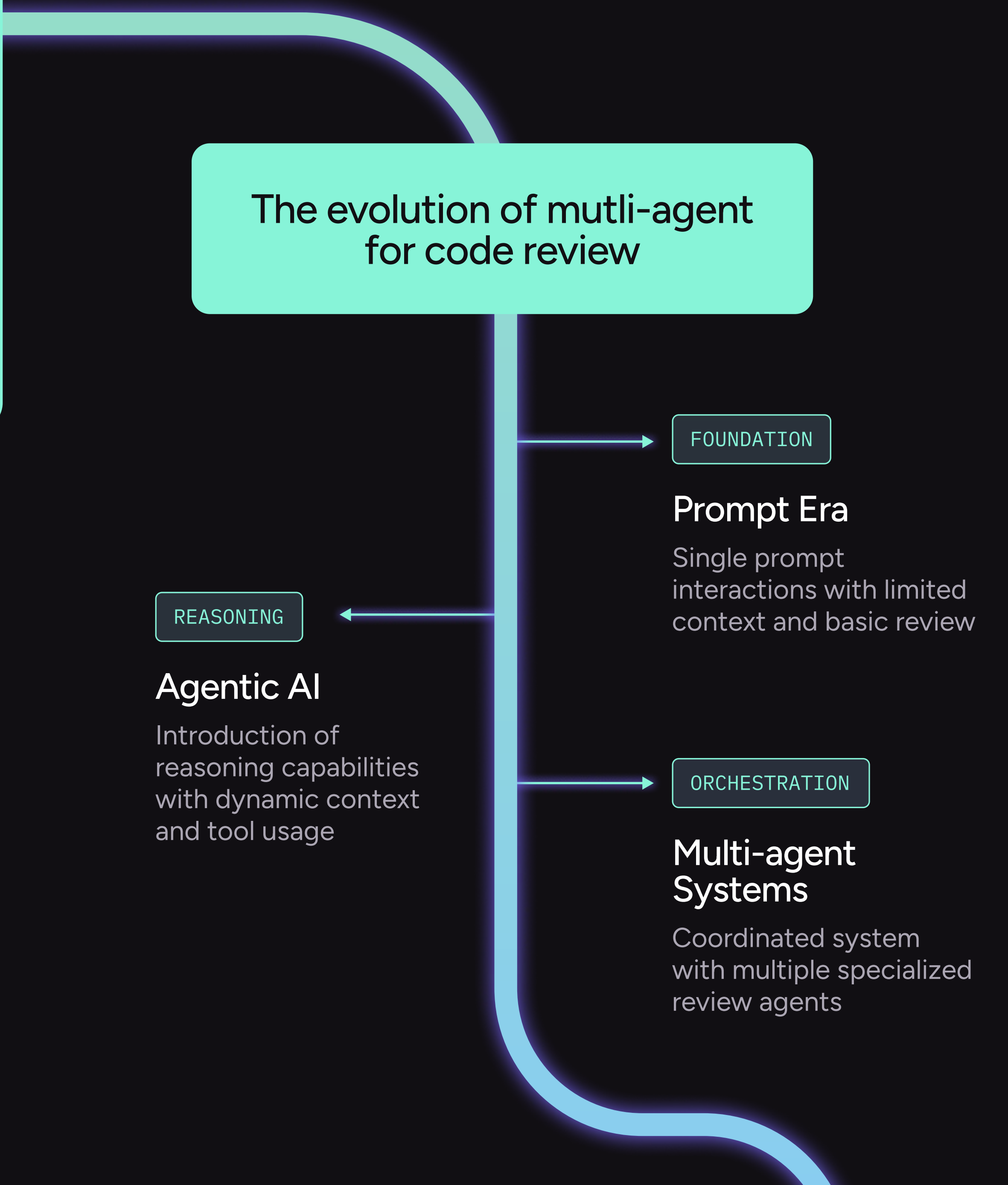
qodo

AI code reviews: the agent swarm approach

A single agent struggles with comprehensive code review because it conflates distinct cognitive tasks. Rather than one generalist model juggling every concern, specialized expert agents operate independently, each focused on a distinct review dimension.

- ✓ Does it work?
- ✓ How does it fail?
- ✓ Is it secure?
- ✓ Do the test prove behavior?
- ✓ Does it perform?
- ✓ Is it maintainable?
- ✓ Is it consistent with coding standards?

Single Agent
Agent Swarm



The evolution of mutli-agent for code review

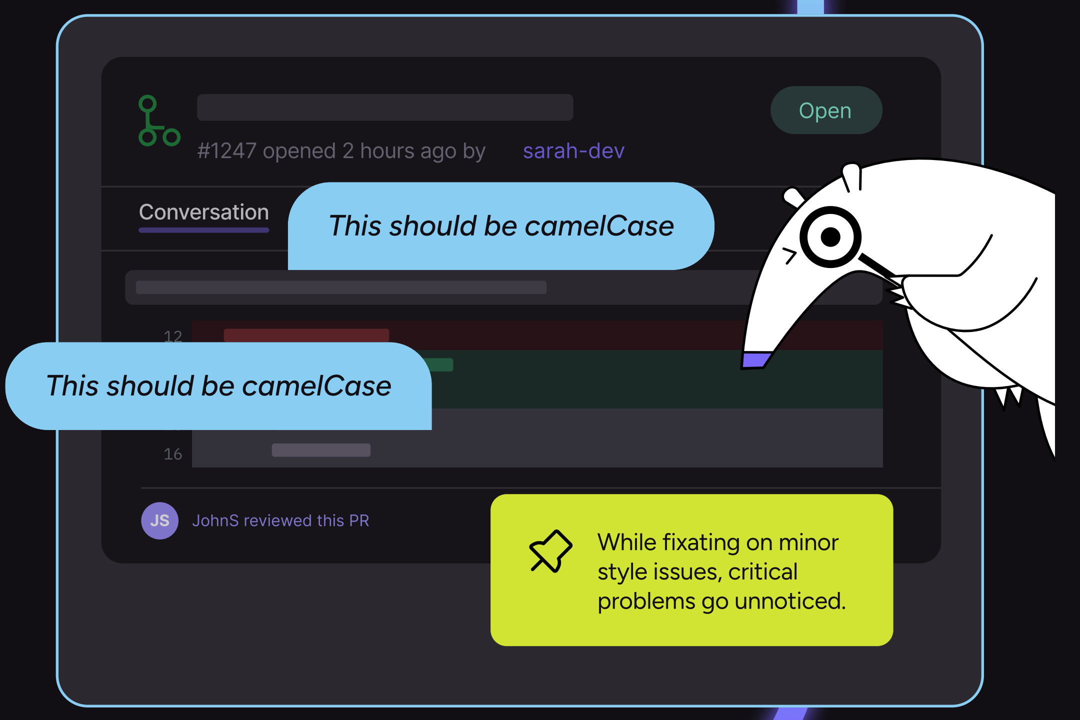
FOUNDATION
Prompt Era
Single prompt interactions with limited context and basic review

REASONING
Agentic AI
Introduction of reasoning capabilities with dynamic context and tool usage

ORCHESTRATION
Multi-agent Systems
Coordinated system with multiple specialized review agents

The Nitpicker

Holds the line on style and consistency

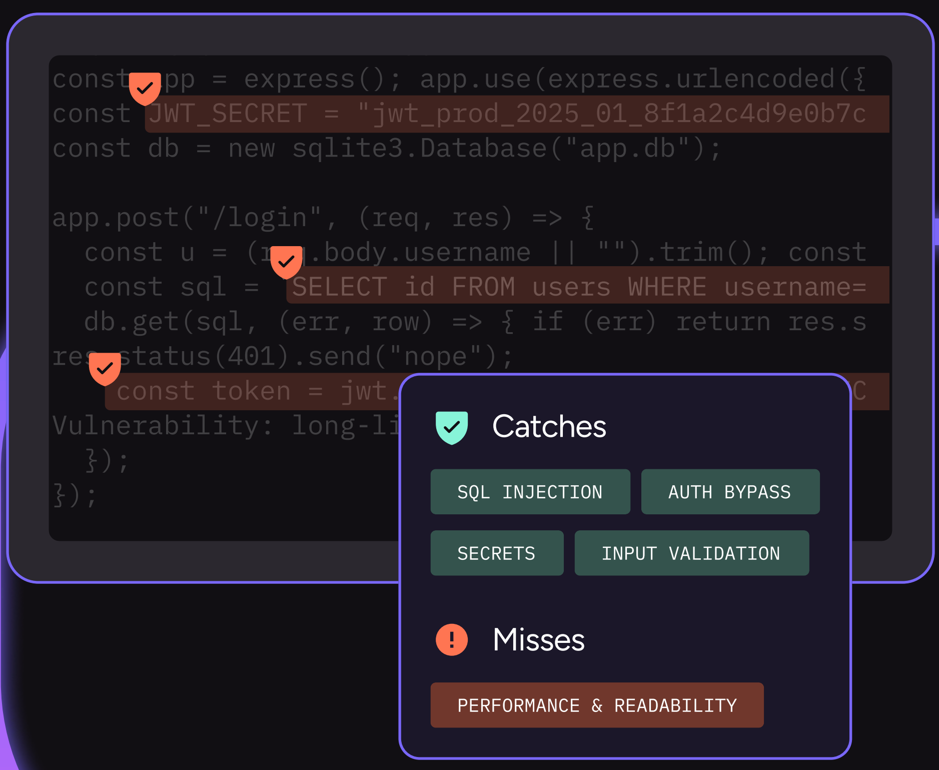


#1247 opened 2 hours ago by sarah-dev
Open
Conversation
This should be camelCase
This should be camelCase
JohnS reviewed this PR

While fixating on minor style issues, critical problems go unnoticed.

The Security Guard

Adversarial thinking to catch exploits

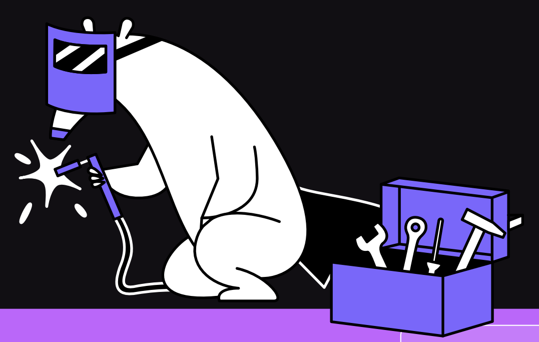


```
const app = express(); app.use(express.urlencoded({
const JWT_SECRET = "jwt_prod_2025_01_8f1a2c4d9e0b7c
const db = new sqlite3.Database("app.db");

app.post("/login", (req, res) => {
  const u = (req.body.username || "").trim(); const
  const sql = SELECT id FROM users WHERE username=
  db.get(sql, (err, row) => { if (err) return res.s
res.status(401).send("nope");
  const token = jwt.
Vulnerability: long-li
  });
});
```

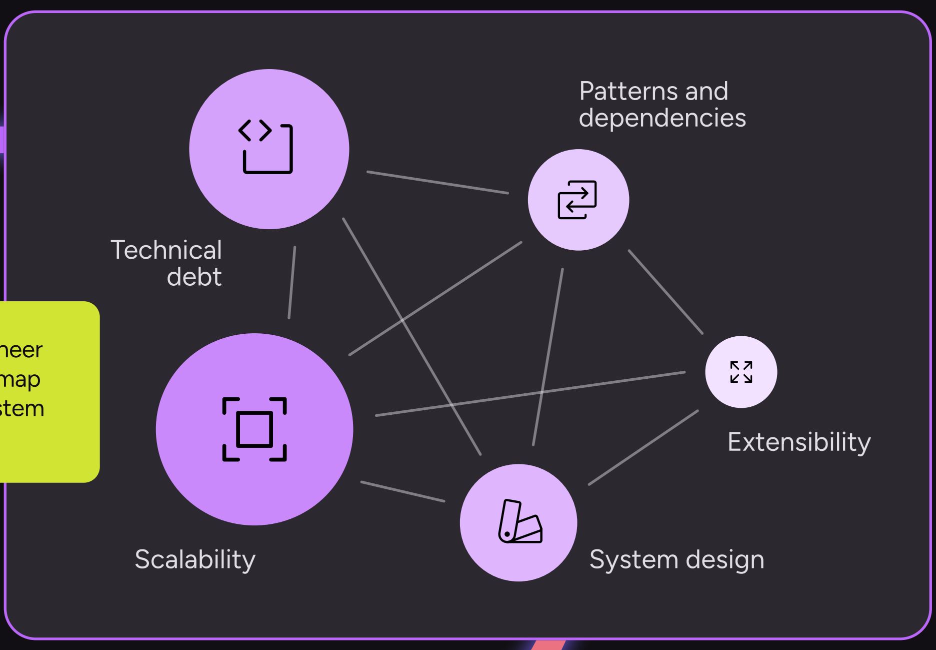
Catches
SQL INJECTION · AUTH BYPASS · SECRETS · INPUT VALIDATION

Misses
PERFORMANCE & READABILITY



The Architect

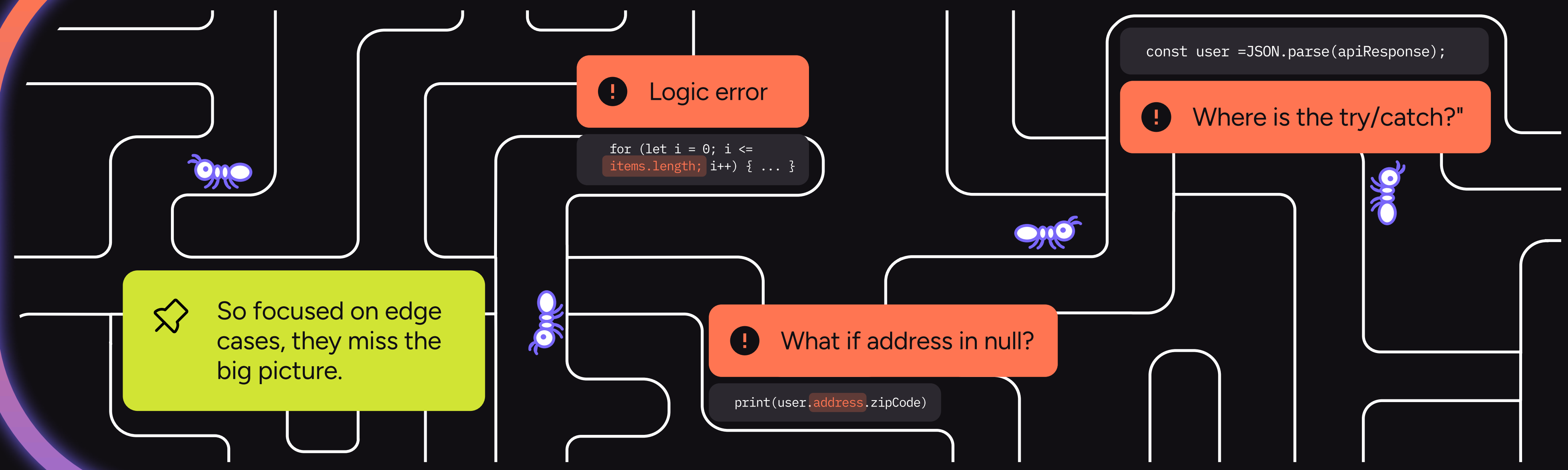
System thinking to prevent technical debt



Usually a senior engineer who holds the mind map of how the entire system fits together.

The Bug Hunter

Catches issues that could be catastrophic



Logic error

```
for (let i = 0; i <= items.length; i++) { ... }
```

```
const user =JSON.parse(apiResponse);
```

Where is the try/catch?"

What if address in null?

```
print(user.address.zipCode)
```

So focused on edge cases, they miss the big picture.

The human reviewer's dilemma

One reviewer cannot maximize depth, speed and coverage all at once.

Something always gives.

Go deep and PRs pile up.

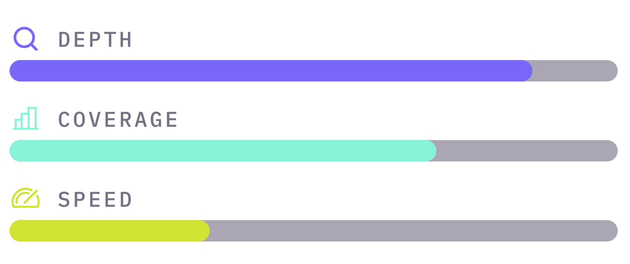
Move fast and issues slip through.

Cover everything and your best reviewers burn out.

Deep + Complete = Slow

PRs wait days. Developers batch changes. Reviewers burn out.

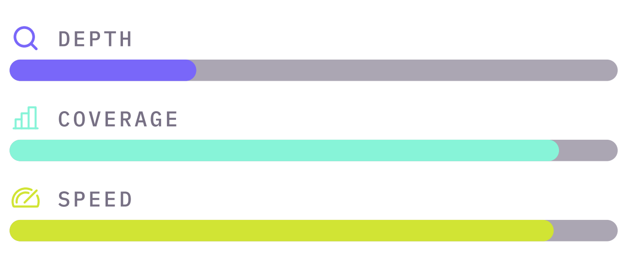
Velocity risk 🟧🟧🟧⬜



Complete + Fast = Shallow

Surface-level review. Security and architecture issues slip through.

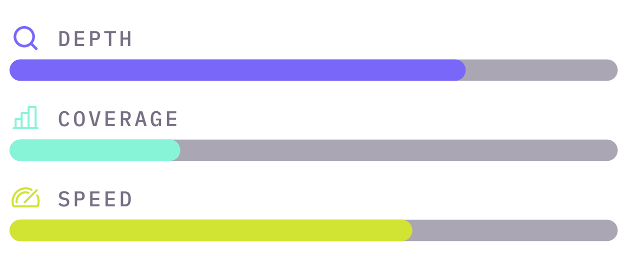
Coverage risk 🟥🟥🟥⬜



Deep + Fast = Selective

Only "important" PRs reviewed. Bugs hide in routine code.

Quality risk 🟥🟥🟥⬜


DEPTH
COVERAGE
SPEED