Not reading code scales

When "humans review the PR" hits its ceiling in the AI era — what comes next.

ai-desk methodology — Hiroyuki OKINOI · 2026-05

1 The common doubt

> **Can a human really read AI-parallel-generated code?**
> **The volume is unmanageable.**
> **But if reviews are fully automated, do you still know what's being built?**

The doubt is correct — **under the assumption that "humans must read the code"**, AI parallel development does not scale.

So what happens if we drop that assumption?

2 The volume wall

The gap between AI generation rate and human reading rate widens every year.

~50	~50,000	×1,000
Lines a human can seriously read per day	Lines AI generates per day	Speed gap (growing, not closing)

The moment "carefully review every PR" stays mandatory, total throughput is **rate-limited by human reading capacity**. Adding more reviewers just means multiple humans reading the same PR.

3 The quality wall — review doesn't catch everything

Even reviewed-by-humans code ships vulnerabilities.

- **Heartbleed (2014)** — OpenSSL, reviewed by experts worldwide, dormant 2 years
- **Log4Shell (2021)** — 8 years dormant, passed Apache Foundation review
- **XZ utils backdoor (2024)** — multi-year planned attack, reached most Linux distros

"Humans review" is **neither necessary nor sufficient**. The basis for confidence is weaker than it feels.

Review is "better than nothing" but not "review = safe". Whether AI or human writes, review alone guarantees little.

4 Re-split the human / machine roles

OLD DIVISION

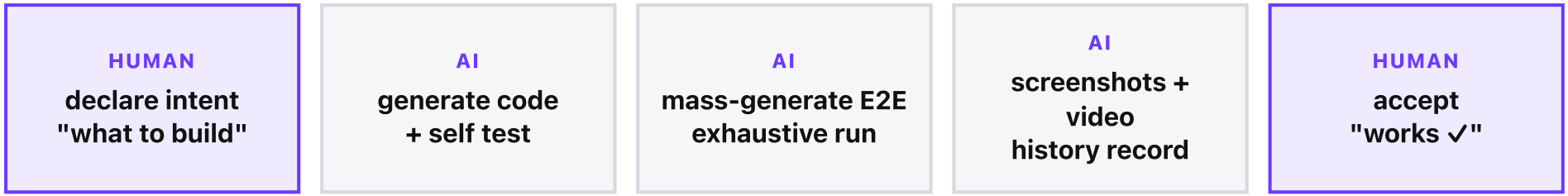
- ✕ Human: writes code
- ✕ Human: reads code (review)
- ✕ Human: writes tests
- ✕ Human: chases bugs
- ✕ Machine: builds & runs

AI-DESK DIVISION

- ✓ Human: **declares intent**
- ✓ Human: **verifies the result** (behaviour / acceptance)
- ✓ Machine (AI): writes / reads / fixes
- ✓ Machine: generates exhaustive E2E
- ✓ Machine: records logs / screenshots / video

What humans give up: chasing lines of code. What humans keep: **"what to build" and "is it working"**.

5 The new pipeline



The human touches only at the start and the end. The 3 middle stages are entirely machine.

Tools shipped in ai-desk:
`ai-desk.js` (code editing) · `ai-eyes.js` (screen observation / input injection / video record) · `eyes-e2e.js` (state → text).

6 Why this scales

Human load becomes O(intent count), no longer dependent on code volume.

OLD (PR REVIEW)

- ✕ Human load = O(lines of code)
- ✕ 1 feature added = hundreds of lines to read
- ✕ 10× features = 10× review hours
- ✕ Parallel dev makes it worse

AI-DESK

- ✓ Human load = O(intent count)
- ✓ 1 feature = 1 intent declaration + 1 acceptance check
- ✓ 10× features = 10× intents (read volume unchanged)
- ✓ Parallel dev: machine side scales, human side stays linear

This is the structure that makes "single-developer full-time parallel execution" work.

7 Objection 1 — "What about future maintenance?"

Q. Code written without reading — surely you can't fix it 6 months later?

A. The fixer is also AI. As long as the original intent and the E2E tests are preserved, AI can re-read and patch. ai-desk's tool for this is **Emblem-scoped editing (skeleton / focus / apply)**. **Even in a 100k-line file, AI only reads the relevant emblem to make the change.**

Premise: AI doesn't go away. With multi-vendor competition + local models maturing, this is structurally guaranteed rather than wishful.

8 Objection 2 — "What about onboarding new members?"

> **Q. A new member who can't read code can't join the project, right?**
>
> **A.** The handoff material = **intent declarations + working E2E tests + screenshots + video**.
> The new member reads "what does this service do", not "how is it implemented", and asks AI for changes.
> Compared with "read the code and infer the spec", **seeing intent and observed behaviour directly is faster onboarding**.

Claim: if it works for an individual, it works for a team — same method, same primitives.

9 Objection 3 — "I can't trust it without reading"

Q. Without reading, you'll never notice when AI does something weird.

A. "Reading" is **subjective inspection**. Sometimes it catches things, sometimes it doesn't (Heartbleed: 2 years dormant).

ai-desk relies on **objective inspection**:

- **Exhaustive E2E** (try all 1920 worlds)
- **Twin (double-entry math)** verification (re-compute GPU output with a pure CPU function)
- **Event sourcing + hash chain** (mathematical tamper detection)

These are **stronger guarantees than "a human read it"**.

10 Real example — fighter-cancel.test.js

Action-game cancel/combo logic, exhaustively tested across 1920 worlds.

1920	19/19	1
Worlds covered (all possible state combinations)	Test cases all green	File (everything in one)

What the human reviews = "all 1920 worlds passed". The contents of individual worlds are not read.

"Mechanically tried everything" is a stronger guarantee than "a human read part of it".

Paradigm shift

Don't give up on "reading the code".

Replace it with what machines can guarantee.

This isn't a "technique improvement", it's a **role-split redesign**.
In an era where AI writes the code, **narrow the human's responsibility to intent and verification**.
The moment that happens, however much code grows, the human's load only grows with intent count.

github.com/AoyamaRito/ai-desk

Hiroyuki OKINOI · Aoyama Rito