コードを読まない開発はスケールする

AI 時代に「人間が PR を読む」が限界に達した時、どこへ行くか

ai-desk 方法論 — 沖井広行 · 2026-05

1 よくある疑問

> **AI 並列開発したコードを、人間が読めるの?**
> **とてもじゃないけど読める量じゃない。**
> **でも AI 任せだと、何作ってるか分からなくなる。**

この疑問は正しい。**「人間がコードを読む」前提のままだと AI 並列開発はスケールしない。**

では前提を変えるとどうなるか。

2 物量の壁

AI の生成速度と人間の読解速度の差が、毎年広がる。

~50
人間が真剣に読める
1日あたり行数

~50,000
AI が生成する
1日あたり行数

×1,000
埋まらない速度差
(さらに広がっている)

「PR を真面目に読む」を維持しようとした瞬間、開発の総スループットは **人間の読解能力で律速** される。チーム拡張しても複数人が同じ PR を読むだけで本質改善しない。

3 質の壁 — 読んでも見つからない

人間がレビューしたコードでも脆弱性は出る。

- **Heartbleed (2014)** — OpenSSL、世界中の専門家がレビュー、2 年潜伏
- **Log4Shell (2021)** — 8 年潜伏、Apache Foundation のレビュー通過
- **XZ utils backdoor (2024)** — 数年に渡る計画的攻撃、Linux ディストリビューション全体に侵入

「人間がレビューする」は **必要条件にも十分条件にもなっていない**。安心感の根拠が弱い。

レビューは「やったほうがいい」けど、「やれば安全」ではない。AI が書こうが人間が書こうが、レビューだけで保証されるものは少ない。

4 人間と機械の役割の再分割

従来の役割分担

- ✕ 人間: コード書く
- ✕ 人間: コード読む (レビュー)
- ✕ 人間: テスト書く
- ✕ 人間: バグ追う
- ✕ 機械: ビルド・実行する

AI-DESK の役割分担

- ✓ 人間: **意図を宣言**する
- ✓ 人間: **結果を検証**する (動作 / 受入条件)
- ✓ 機械 (AI): コード書く・読む・直す
- ✓ 機械: テスト大量生成 (E2E 網羅)
- ✓ 機械: 実行ログ・スクショ・動画

人間が手放すのは「コード行を目で追う作業」、保つのは「何を作りたいか・何が動けば成功か」の判断。

5 新しいパイプライン

人間	AI	AI	AI	人間
意図宣言 「何を作る」	コード生成 + 自己テスト	E2E 大量生成 網羅テスト実行	スクショ + 動画 履歴記録	受入確認 「動いてる ✓」

人間の介入点は 最初と最後の 2 箇所だけ。間の 3 つは全て AI / 機械。

ai-desk 上の道具:
`ai-desk.js` (コード操作) ・ `ai-eyes.js` (画面観測 / 入力流し込み / 動画記録) ・ `eyes-e2e.js` (状態 → text 変換)。

6 なぜスケールするか

人間の負荷が 意図の数 に比例し、コード量に依存しなくなる。

従来 (PR レビュー)

- ✕ 人間負荷 = O(コード行数)
- ✕ 機能 1 個追加 = 数百行 read
- ✕ 10 倍の機能 = 10 倍のレビュー時間
- ✕ 並列開発が増えるほど詰む

AI-DESK

- ✓ 人間負荷 = O(意図数)
- ✓ 機能 1 個追加 = 1 つの意図宣言 + 1 つの受入確認
- ✓ 10 倍の機能 = 10 倍の意図 (read 量は変わらず)
- ✓ 並列開発で機械側が比例増、人間側は線形増のまま

これが「個人開発でフルタイム並列実行」を成立させる構造。

7 反論 1 — 「将来の保守どうする?」

Q. 読まずに作ったコード、半年後に直せないだろう?

A. 直すのも AI。意図 (元の宣言) と E2E テストが残っている限り、AI が読み直してパッチを当てられる。
ai-desk の **Emblem 単位の編集 (skeleton / focus / apply)** がこのための道具。 **10 万行のファイルでも該当 emblem だけ AI が読めば修正できる。**

前提: AI が止まらないこと。これは API の問題ではなく、複数 AI ベンダー競合 + ローカルモデル普及で構造的に保証されつつある。

8 反論２— 「メンバー追加 / 引継ぎは?」

Q. 新メンバーが入った時、コード読めないと参加できないだろう?

A. 引継ぎ資料 = **意図宣言 + 動く E2E テスト + スクショ + 動画**。
新メンバーは「コード」ではなく「**このサービスは何をするか**」を読み、AI に修正を依頼する。
従来の「コード読んで内部仕様を察する」より、**意図と実際の挙動が直接見える方が引継ぎは早い**。

これは「個人開発が成立するなら、複数人開発も同じ方法で成立する」という主張。チームスケールは別問題に見えるが、実は同じ問題。

9 反論３— 「信頼できない」

Q. 読まないと、AI が何か変なことしても気付けない

A. 「読む」は **主観的検査**。気付くこともあれば気付かないこともある (Heartbleed 2 年潜伏)。
ai-desk が頼るのは **客観的検査**:
・**網羅 E2E** (1920 世界全部試す)
・**Twin による複式数学検算** (GPU 出力を CPU 純粋関数で再計算)
・**イベントソーシング + ハッシュ連鎖** (改ざん検知)
これらは「人間が読む」より **確実な信頼根拠**。

10 実例 — fighter-cancel.test.js

アクションゲームのキャンセル・コンボ判定を 1920 世界全部 網羅テスト。

1920	19/19	1
テスト対象世界 (全可能状態組合せ)	テストケース 全合格	ファイル (全部 1 ファイルに集約)

人間がレビューする規模 = 「1920 世界が全部 OK だった」という結果だけ。
個別の世界の中身は読まない。**機械的に「全部試した」が保証されている方が、人間が一部読むより強い保証。**

パラダイムが変わる

「コードを読む」を諦めるのではなく、
機械的に保証できるものに置き換える。

これは「テクニックの改善」ではなく「**役割分担の再設計**」。
AI が書く時代に合わせて、**人間の責任範囲を意図と検証に絞る。**
その瞬間、コード量がいくら増えても、人間の負荷は意図の数しか増えない。

github.com/AoyamaRito/ai-desk

沖井広行 · 蒼山りと