

PLAYBOOK · FREE TO READ

# The AI SDLC Playbook

AI writes more of your code every month and you are still the one who has to read all of it. Two ceilings and six practices for what to do about that, all of it from running an AI-operated delivery process on a live product past 165,000 lines as at 2 August 2026 — including the parts that went badly, which are the useful parts.

No email required. Nothing gated. If you take all of it and never speak to us, the playbook has done its job.

## Contents

- 1 The two ceilings
- 2 Green tests can sit over a broken product
- 3 Gates must tighten as the codebase grows
- 4 Idle is a status to report, not hide
- 5 Blocked should be the loudest thing on screen
- 6 Recompute metrics, do not remember them
- 7 Multiply reviewers, not the reviewer
- 8 What to adopt on Monday

## 1 The two ceilings

Coding was never the only process in building software, just the one that got faster first. Requirements, design, security review, testing and release are most of the real work, and

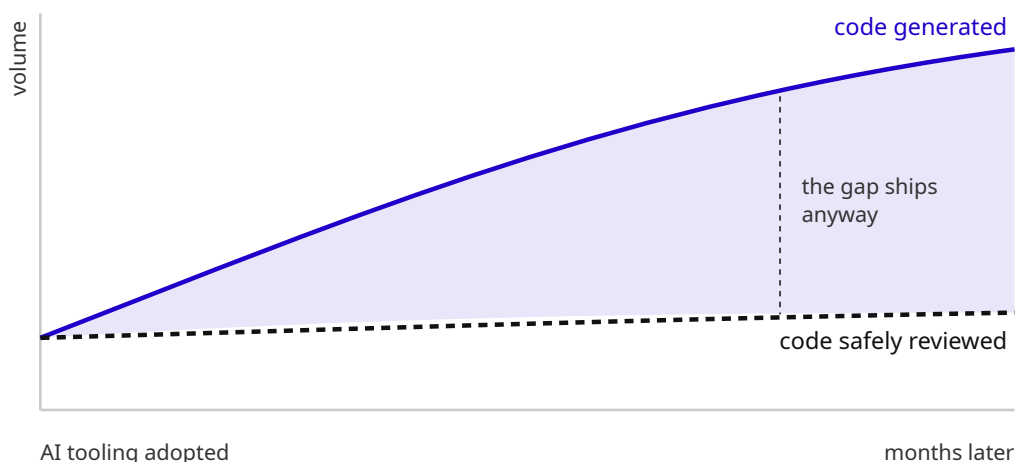
none of them sped up because generation did. Point AI at “write the feature” without touching the rest of the lifecycle and you have handed a slower pipeline more raw material than it can absorb.

### Your reviewer is your throughput

AI can 10× how much code gets written. It cannot 10× how much a human can safely review. If every line still passes through one person's eyes, that person's reading speed is your real ceiling — no matter how fast the model is.

### It degrades exactly where you are going

AI is excellent on day one: a blank file and an afternoon get you a week of senior work. Real products cross into six figures of lines, where parts depend on each other in ways nobody wrote down. The model does not get worse — the blast radius of confidently wrong grows with the codebase, and nothing tells you unless something is built to.



The gap is not idle work. It is code that ships having been read less carefully than the code before it. We think that is what the 1.7× figure is measuring, though the study reports the association and not the cause.

Those were written in July 2026. The industry has since measured the same thing:

- **1.7×** more issues in pull requests containing AI-assisted code than in human-written code — [arXiv preprint 2603.28592, March 2026](https://arxiv.org/html/2603.28592v2) (<https://arxiv.org/html/2603.28592v2>)

- **30–41%** rise in technical debt within six months of widespread AI tool adoption — Innovative Group, 2026 (<https://innovativegroup.io/blog/ai-code-technical-debt/>)
- **45%** of AI-generated code introduced a security vulnerability — Innovative Group, 2026 (<https://innovativegroup.io/blog/ai-code-technical-debt/>)
- **4×** maintenance costs by year two, for teams that do not actively manage the debt — Innovative Group, 2026 (<https://innovativegroup.io/blog/ai-code-technical-debt/>)

Published figures, directional rather than audited. Three of the four come from the same 2026 analysis; the first is a separate preprint. Every one is linked so you can weigh it yourself.

## 2 Green tests can sit over a broken product

Our test suite passed. Coverage was at 80%, the figure on [our published board as at 2 August 2026](#). Every gate reported success — and two screens were failing to load, with exactly one of nine customer journeys completable end to end.

That is not a criticism of testing; it is the definition of it. A test encodes an expectation, and an expectation is something a person had. The failures that hurt are the ones nobody expected, which is precisely the set no test covers. More tests push the boundary outward, and the boundary never closes.

The practice: add one check that asks whether a real person can finish the job on the running system, and let it block a release.

It is the one signal that cannot be satisfied by writing more code to satisfy it. If you do nothing else on this list, do this.

## 3 Gates must tighten as the codebase grows

What was good enough at 5,000 lines is not good enough at 165,000. The number of ways a system can be broken grows faster than the system does, so a fixed quality bar is a falling one — and it fails quietly, which is the worst way to fail.

The practice: make at least one threshold a function of size.

Coverage requirements, review depth, how much of the system a change may touch without extra scrutiny. This is unpopular with anyone who wants a stable definition of done, and it is the only version that stays honest as the product grows.

## 4 Idle is a status to report, not hide

Every delivery process has idle time. Processes that hide it do not have less of it; they have the same amount plus a reporting layer that obscures where it went.

The practice: put “idle, zero done” on the board in plain text when it is true.

It changes behaviour faster than any stand-up, because it is visible without anyone having to raise it — and raising it is exactly what people avoid doing. Ours records 28% of elapsed time as time actually spent working, as at 2 August 2026, against a commonly published 10–20%. Read the flattering way, that is above the range. Read honestly, roughly three-quarters of the clock is still waiting, and we only know which quarter is which because the number is on [the board rather than in a report](#).

## 5 Blocked should be the loudest thing on screen

Blocked work is the most expensive state in any pipeline and usually the least visible. It surfaces at the next stand-up, or in a status report, or three weeks later when someone asks why a feature never shipped.

The practice: blocked work shows the day it blocks, at the top, before anything green.

## 6 Recompute metrics, do not remember them

A number updated last Friday is a story, not a fact. Any metric maintained by hand drifts from reality at exactly the rate that makes it dangerous: slowly enough to keep trusting, fast enough to be wrong when it matters.

The practice: derive every number from the work record, and show the ones that do not flatter you.

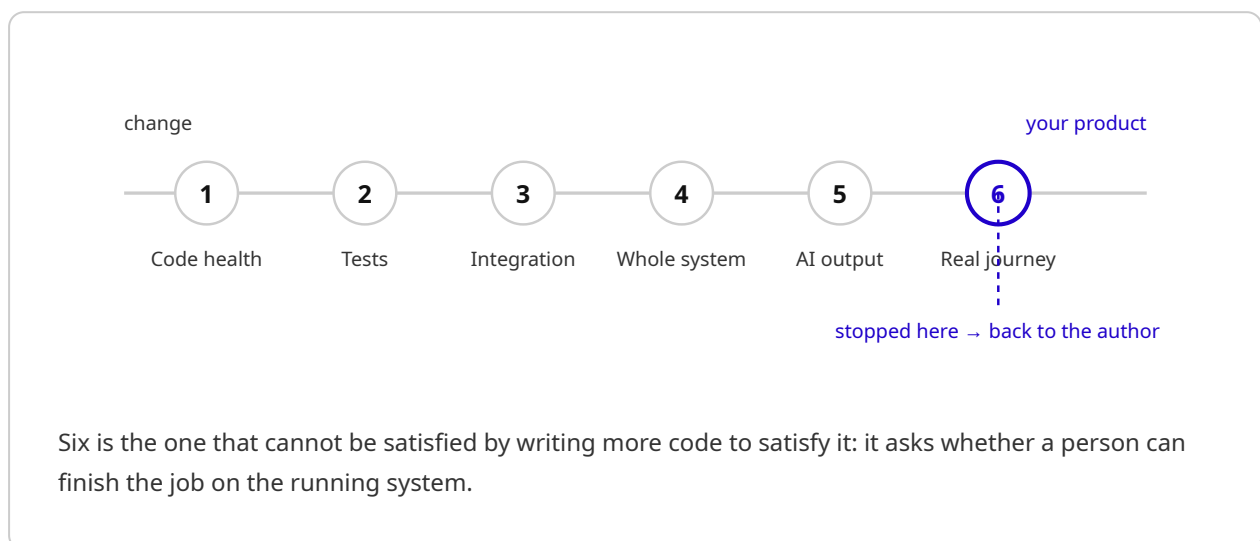
Ours, on the position as at 2 August 2026, [reports that 43% of work was later redone](#) — a figure we publish with the caveat that there is no published industry norm to compare it against, so treat it as our own watch item and not as a benchmark you have beaten. The same board shows queueing time rising as more work ran in parallel; we can show the direction and not a magnitude we would defend, so take it as a direction. Publishing both is not humility, it is self-interest: a board that can only show good news tells you nothing on the day something goes wrong, which is the only day you urgently needed it.

## 7 Multiply reviewers, not the reviewer

This is the answer to the first ceiling. If every line passes through one person, that person is your throughput — and hiring another of them is slow, expensive, and the reason you reached for AI in the first place.

The practice: build a ladder. Automated checks that block on failure, then narrow human judgement only where judgement is genuinely required.

Ours is six checks: code health and secrets, tests, integration, a full rebuild with a vulnerability scan, a sampled score of what the AI actually produced, and finally whether a person can finish the job. A human still decides — on a much smaller surface.



## 8 What to adopt on Monday

Cheapest first. The first two cost an afternoon between them.

1. **Put blocked at the top of your board.** A sort order change. Costs nothing.
2. **Report idle honestly for two weeks.** Do not act on it yet. Just look at it.
3. **Pick one number and recompute it from source.** Whichever one you would be most embarrassed to find is wrong.
4. **Add one end-to-end journey check** for the single path that matters most, and let it block a release.
5. **Sample AI-written changes and score them.** Ten a week is enough to see a trend.
6. **Tie one threshold to codebase size** so the bar moves without anyone having to argue for it.

None of this requires our tooling, and we would rather you did it yourself than hired anyone. The one thing worth reading next is what happens when a team actually runs all six: [our own board, as at 2 August 2026, with the rework and queueing rows left in](#). It is the only page here that is mostly numbers, including the ones we would rather not print.