
Software Reliability
AI Readiness Report
Reliably.

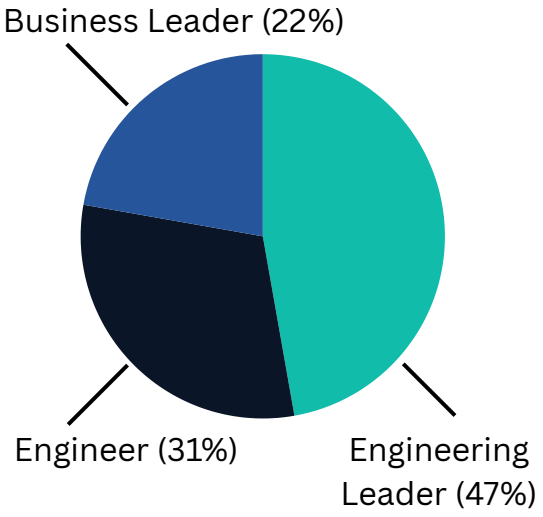
84% of engineering teams have already had an AI-related incident, or admit it's only a matter of time.

Executive Summary

Confidence in AI-generated code has outpaced the practices needed to verify it.

- 64% say reliability has stayed the same or gotten worse since adopting AI.
- 58% aren't tracking AI-generated work separately from human work at all.
- 79% are confident in shipping AI-generated code despite the tracking gap.
- Nearly one in three teams would face a painful recovery, or don't know what would happen at all, if their AI systems broke.

Who We Asked



Distribution of Participant Roles (N=36)

Responses came from individuals who build software for companies ranging from under 10 to over 1000 employees. Questions were focused on the reliability of their software systems after integrating with AI tooling.

Percentages reflect the subset of respondents who answered each individual question.

Written by Tyler Desplenter, PhD · ©2026 Reliably

MYTH 01

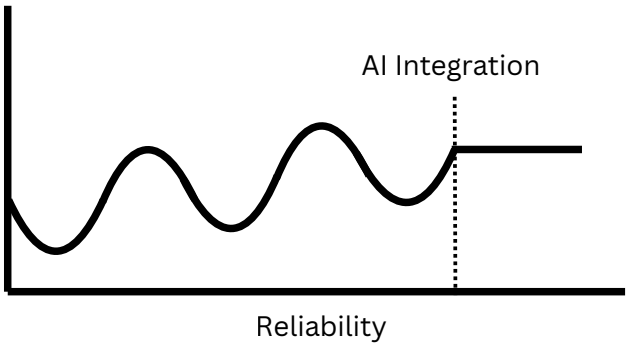
> “Clearing tech debt with AI should mean fewer things break.”

THE REALITY

64% of participants say that system reliability is flat or has gotten worse since integrating with AI.

Paying off tech debt is possible with the help of AI-assisted development. AI tooling lets teams produce changes at the fastest rate they've ever experienced. That speed means finally clearing the backlog nobody ever had time for, and finally getting to **the infrastructure projects that actually improve reliability**. It's reasonable to expect that speeding up those initiatives with AI would show up as measurable improvement. For some teams, reliability was reported as improving since AI integration. The old code got rewritten. The outdated dependencies got upgraded. The tickets that had sat untouched for years finally moved.

But the majority of participants report neutral or negative results instead. Clearing debt and improving reliability aren't automatically the same thing. It only counts if what replaces the old code actually holds up under real conditions. Even a steady rate of defects looks different at a much larger volume of changes. The same defect rate across more total changes means the absolute number of defects has gone up, not down. More software is getting shipped, but for most teams reliability has stalled, and for some, it's gotten worse.



"A significant portion of the code was generated with AI. When developers revisit the code later, they may not fully understand the reasoning behind certain implementation decisions. They can explain what the code does, but not why a particular approach was chosen."

Question: "Since integrating AI into your development processes, has your system's overall reliability improved, gotten worse, or stayed the same?" (N=33)

MYTH 02

“We test everything AI touches, so we’re covered”

THE REALITY

42% of teams have already had an AI-related production incident.

"We test everything AI touches" is a reasonable thing to believe if review was built for a slower, lower volume world. Most engineering teams already have some form of code review or quality assurance, a gate meant to catch a bad change before it reaches production. That gate worked well enough for years, when a human wrote most of the code and the pace of change matched what a human reviewer could keep up with. AI didn't remove that gate. It just changed how much has to move through it.

For a lot of teams, the gate didn't scale with the volume. One respondent described QA coming under more pressure than it could handle, leading to what they called rubber stamping. Another said engineers simply don't have the capacity to properly review code changes at this rate. And when something does get through, the failure isn't always a clean one. Of the teams that had an incident, about one in five couldn't determine the root cause at all. Somewhere between "we tested it" and "it broke in production" is a much larger group who reviewed the code, missed the problem, and in some cases still don't know why.

INCIDENT REPORT

Root cause:
AI-generated code

“The more access you give an AI agent the more useful it can become. On the other hand it introduces significant risk to your infrastructure if left unchecked.”

Question: "Have you experienced a production incident caused by AI-generated code or an AI pipeline failure?" (N=33)

MYTH 03

“We’re confident in the AI-generated code that we ship.”

THE REALITY

55% of teams say they're confident shipping AI-generated code, but admit the gaps are still there.

Most teams report being confident shipping AI-generated code to production. Look closer, though, and the confidence isn't quite what it sounds like. The largest single group didn't say they were fully confident. They said they were somewhat confident, and specifically noted that they know gaps exist. That's a different claim than "we've validated this." It's closer to "we're shipping it anyway, aware that we might be wrong."

That confidence has to be resting on something, and the most obvious candidate is the review process teams already trust to catch problems. The previous results showed how much weight that process is actually carrying. A meaningful share of teams have already had an AI-related incident, and some of those couldn't even determine the root cause afterward. If review were consistently catching what it needed to catch, that picture would look very different. Confidence in AI-generated code isn't unfounded, exactly. It's just outpacing the practices that would actually earn it.

"Blind faith with no timeline adjustment to handle literally hundreds of new defects without root cause, investigation time, or appropriate risk analysis."

Question: "How confident is your team in the reliability of AI-generated code before it ships to production?" (N=33)

AI tooling can be unreliable itself.

"AI forgets important design and architectural constraints, leading to rework on things we thought were 'done.'" — Founder, 1–10 employees

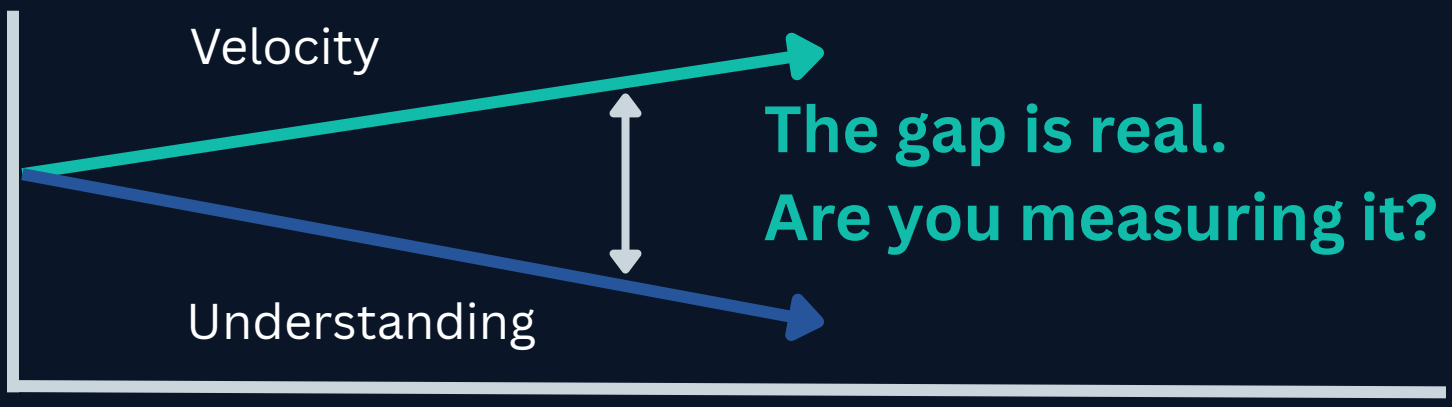
---

And velocity is outpacing understanding.

"The velocity and volume of code generated makes it impossible to review every single line thoroughly."
— Engineering Manager, 51–200 employees

"Non-deterministic outcomes lead to complex debugging. AI at every level makes it harder to understand large volumes of code."
— Staff Quality Engineer, 201–500 employees

"We're pushing code we don't fully understand."
— Engineering Manager, 1,000+ employees



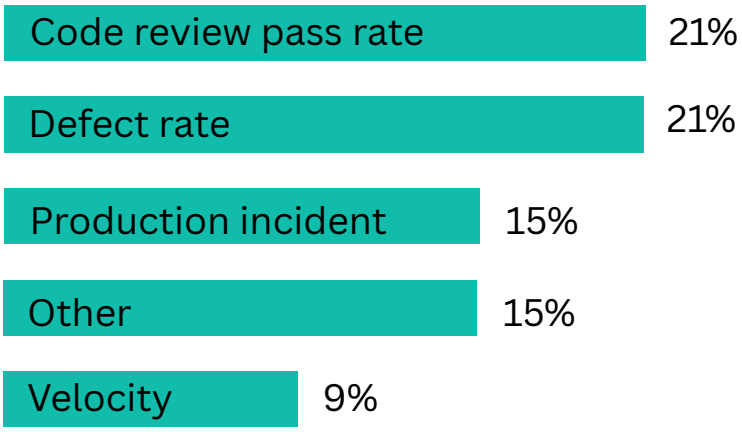
Knowing your AI tooling is reliable starts with measuring it.

But 58% of teams aren't tracking AI-generated work separately from human work at all.

Most teams assume they'd know if AI-generated work was causing problems. That assumption feels safe mostly because it's never been tested. There's usually some kind of signal when something's badly wrong, a build fails, a customer complains, a page goes off at midnight. That feels like enough. But visible failure isn't the same thing as real visibility. Just because nothing's obviously on fire doesn't mean the system underneath is actually accounted for.

58% of teams aren't tracking AI-generated work separately from human work at all. The previous results already showed what that costs when something breaks. About one in five teams that had an incident couldn't determine the root cause. They didn't lack a signal, they lacked the ability to trace it back to its source. Waiting for something to visibly fail isn't the same as knowing where it came from. For a meaningful share of teams, that difference only became obvious after the fact.

Metrics Attributed to AI Deliverables



> "Engineers sometimes rely too heavily on AI-generated code and merge changes as long as the basic functionality appears to work. The implementation may not account for all scenarios, which can lead to future failures, especially when test coverage is missing."

Question: "How do you track AI deliverables versus human deliverables?" (N=33)

Confidence is high despite AI-related production incidents, stagnant reliability, and a lack of tracking AI deliverables.

What Does This Mean?

Teams are not ready for the reliability challenges AI introduces, and they're moving forward anyway. The pressure to adopt is outpacing the pressure to understand what it's actually doing to the systems it touches. Reliability was expected to improve. For most teams, it stalled or slipped instead. Incidents followed the same shape, review didn't scale with the volume moving through it. Confidence stayed high through both.

Tracking is the thread that ties those together. Most teams aren't measuring how much of their system AI has actually touched, so there's little feeding back into that confidence to test whether it's still accurate. Reliability, incidents, and confidence should move together. Without tracking, there's less to notice when they don't.

Most teams have indicated that they aren’t ready to deal with the software reliability challenges of integrating with AI.

Every finding in this report traces back to the same gap: measurement. A team that can't say how much of its system AI has touched, or why an incident happened, can't actually manage the risk it's already taken on. AI-generated work has been shipping alongside everything else, unmeasured and unverified on its own terms, moving through the same pipelines and the same review as everything written by hand. Nothing distinguishes its track record from theirs. The confidence teams report is real, it's just resting on a foundation that hasn't been inspected yet.

Failures due to AI are happening.
Reliability means being able to recover.

Nearly one in three teams would face a painful recovery, or don't know what would happen at all.

Being ready for an AI failure starts with knowing how fast you'd actually recover, and not every team clears that bar. Nearly one in three teams would face a painful, days-long recovery, or genuinely don't know what would happen at all, because they've never tested it. That's not a hypothetical gap. It's the practical cost of everything else in this report, a team that isn't tracking what AI has touched can't say with any confidence how quickly it could untangle a failure if one happened.

The more reassuring numbers deserve a second look too. 62% say they could recover in minutes or hours, but that range hides a real difference in what recovery actually looks like. Minutes, with a genuine fallback in place, is recovery in the way most people mean it. Hours, with a manual workaround, usually means someone keeping things running by hand, and a full day of degraded service is still a full day customers can feel, whether or not it ever shows up as a logged incident. Somewhere between recovered and still recovering, a meaningful share of teams are doing the second one and calling it the first.

> *If your AI tooling broke right now, how quickly could you recover, and how confident are you in that number?*

Question: "If your AI tools broke right now, how quickly could you recover and keep serving your customers?" (N=29)

You’re moving faster with AI, without understanding the cost.

You can’t manage what you don’t measure.

Start measuring.

reliably.ca/#assessment

Reliably.