

AI Coding by the Numbers: 42% of Committed Code, 96% Who Don’t Trust It, 48% Who Check

Sonar’s 2026 survey says 42% of committed code is now written or significantly assisted by AI. The same survey says 96% of developers do not fully trust that code to be functionally correct.

THE SHORT VERSION

Sonar’s 2026 State of Code Developer Survey, published November 4, 2026 and based on 1,149 professional developers, found that 42% of committed code is AI-generated or AI-assisted, 96% of developers do not fully trust AI code to be functionally correct, and only 48% always verify it before committing. Distrust is close to universal. Verification is a coin flip.

The 48-point gap between distrusting AI code and checking it

One number reframes everything else in the dataset: 48. That is the spread between the 96% who say they do not fully trust AI-generated code and the 48% who say they always check it before it goes in. Half the developers who distrust the output ship it anyway, at least some of the time.

Developers are not confused about the reliability of these tools. Sonar's ROI analysis puts distrust at the kind of level you normally only see on questions where everyone answers the same way out of politeness. The verification rate still sits at half. Whatever is stopping people from checking, it is not ignorance.

42%
of committed code generated or significantly assisted by AI
Sonar 2026 State of Code, 1,149 developers, published Nov 4, 2026

96%
do not fully trust AI code to be functionally correct
Sonar AI coding ROI analysis, Nov 6, 2026

48%
always check AI-assisted code before committing
Sonar AI coding ROI analysis, Nov 6, 2026

65%
expected AI share of committed code by 2027
Sonar press release, November 2026

My read: the gap exists because verification is unpriced. Generation has a visible cost (a subscription line item) and a visible benefit (a diff appearing in seconds). Review has an invisible cost paid in attention, with no metric attached to it at all. When one side of a trade is measured and the other is not, the measured side wins. That is an organizational design fact rather than a character flaw in engineers.

Why so much AI code passes review when developers don't trust it

Because it reads well. 61% of respondents say AI often produces code that looks correct

but is unreliable. That is the mechanism. Human code that is wrong usually looks wrong in some small way: an odd variable name, a missing guard, a function that quietly does too much. AI code is stylistically clean and structurally plausible, so the cheap heuristics reviewers have built over a career stop firing.

The downstream number confirms it. 53% report that apparently-correct but unreliable AI code has hurt their technical debt. That is a reported outcome from more than half the sample, not a theoretical risk.

Reviewing this code is also harder work. 38% say reviewing AI-generated code takes more effort than reviewing a colleague's code, and the split by seniority is the interesting part: 40% of juniors versus 29% of seniors. I would have guessed the opposite, with experienced engineers more sensitive to the extra effort because they know what proper review costs. The data says juniors feel the burden more. One explanation fits: reviewing code whose author cannot explain their reasoning is a skill you build by writing a lot of code yourself, and juniors have not built it yet.

> ! **Read the fieldwork date before you quote this**
> Sonar published on November 4, 2026, but the fieldwork ran throughout October 2025. These answers are roughly a year old at publication. The sample is described as global with no published country breakdown, no quotas and no margin of error. n=1,149, self-reported, vendor-run, and Sonar sells code quality tooling. The directions are credible. Treat the decimal places as decoration.

How the 42% stacks up against Stack Overflow and DX

The headline share is roughly consistent across independent datasets, which matters more than any single figure. Sonar reports 42% of committed code. DX's Q2 2026 report across 500+ engineering organizations put AI at 52% of merged code. Stack Overflow's 2026 Developer Survey, published October 6, 2026 with more than 30,000 technologists across 169 countries, found 83% using at least one AI tool and 65.9% using coding assistants or agents.

Those measure different things (share of code versus share of developers, committed versus merged, October 2025 fieldwork versus later), so they are not directly comparable.

What they agree on is the regime. AI authorship of production code is the normal case now, somewhere in the 40 to 55 percent band depending on how you count, with adoption among individual developers well above 60%.

Adoption among those who try it is close to total: 72% of developers who have tried AI coding tools now use them daily. That is the stickiness number, and it tells you the 42% is not going to drift back down. Sonar's respondents expect 65% by 2027. My take: surveys asking people to predict their own future behaviour are soft evidence, but the direction is not in doubt, and 65% in 2027 would mean most of the code in a repository was authored by something that cannot be asked why it did that.

A third of developers reach AI tools through personal accounts

One figure in the release gets less attention than it deserves. 35% of developers access AI coding tools through personal accounts rather than work-sanctioned ones. Over a third of AI-assisted code entering commercial repositories came through a channel the employer did not provision, cannot audit and in most cases does not know about.

Two problems follow directly. There is no visibility into what model produced what, and no control over whether proprietary code left the building as context in a prompt. An organization that cannot say which tools touched its codebase cannot give a clean answer when someone asks.

Sonar's release borrows Werner Vogels' term **verification debt** for the accumulating pile of unchecked AI output. I think the framing is right and slightly too gentle. Technical debt is code you know is bad and chose to ship. Verification debt is code where nobody knows, and the not-knowing compounds quietly, because the next AI-generated change is written against a codebase whose invariants were never confirmed in the first place. I covered the structural side of this in the 623-million-change analysis showing refactoring down 70% and duplication up 81%: volume rising, cleanup falling.

A third of teams tightened their controls, a third are still deciding

Sonar reports that roughly one-third of development teams say they are now more rigorous about code security, quality and compliance because of AI, and roughly another third are still evaluating what to change. The report does not break out what the remaining third said, so I will not guess.

Note the shape, though. A year into mass adoption of AI coding tools, at most a third of teams have adjusted their controls. The tooling moved fast and the review process did not follow in most organizations, and the people doing the reviewing know it. That is a large part of why 96% say they do not fully trust what they are committing.

WHAT I WOULD DO

Stop trying to lift the 48% by exhortation. Nobody reviews more carefully because a policy asks them to. Make AI-assisted commits machine-identifiable so you can measure defect rates by origin instead of arguing about them, and move the heaviest review burden off juniors, since the 40% versus 29% split says they are carrying the cost of a skill they have not built yet. Then close the 35% personal-account gap, because you cannot govern a tool you did not know was in the building. Unconfirmed, my reading only: I expect the next round of this survey to show verification rates falling rather than rising, because the volume of generated code grows faster than review capacity and nothing in the current incentive structure corrects that.

What these numbers license you to say, and what they don't

Confirmed by the data: AI writes or assists close to half of committed code, nearly all developers distrust it, about half always check it, and more than half have seen it damage their technical debt. Not confirmed: whether the untrusted code is causing more production incidents. Sonar measured perception and self-reported practice, not defect escape rates. That is the study I want and have not seen.

Worth saying plainly as well: 42% is self-reported. Developers are estimating what share of their committed code the AI wrote, in a survey, a year ago. The DX figure of 52% comes from instrumented engineering data and is higher, which is roughly what you would expect if people underestimate how much of the boilerplate was not theirs.

If you run engineering and you cannot currently answer "what percentage of our merged code was AI-authored, and what is our defect rate on it versus human-authored code", you are in the two-thirds of teams that have not adjusted. That is a measurement problem before it is a quality problem, and measurement problems are the cheap ones to fix. If that is where you are and you want a second opinion on where to start, I am happy to talk it through.



WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →